

DESIGN AND IMPLEMENTATION OF PRINT SERVER ON IBM PC/XT FOR STAR NETWORK

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

PRASAD V. BHATT

to the

**DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR**

FEBRUARY, 1989

CERTIFICATE

It is certified that this work 'DESIGN AND IMPLEMENTATION OF PRINT SERVER ON IBM PC/XT FOR STAR NETWORK' by Mr. Prasad V. Bhatt has been carried out under my supervision and this has not been submitted elsewhere for a degree.



(A. Joshi)
Professor

Department of Electrical Engineering
Indian Institute of Technology
Kanpur.

V-901

-4 OCT 1989

LIBRARY

105876

Th
621-38195
B 469d

EE-1989-M-BHA-DES

ACKNOWLEDGEMENT

I express my deepest sense of gratitude to Dr. A. Joshi for his able guidance during the course of the present work. I also thank him for the advice given when the hardware had lot of problems.

My thanks to Mr. Arun Kanabar, Mr. G.N.M Sudhakar, Mr. R.Gopalkrishna, Mr. George Joy, Mr. S.S. Bhatnagar and other R.E.s of Microprocessor Development Systems Lab for the help provided during the project work.

I appreciate all my friends for valuable suggestions provided by them. My special thanks to Mr. Deepak Murthy and Mr. I. Ravi for the innumerable help given during the course of this present work.

I will be failing in my duties if I do not express thanks to my Uncle Mr. E. Narayanan for acting as my benefactor during my stay over here.

Prasad. V. Bhatt

TABLE OF CONTENTS

Chapter 1 : INTRODUCTION

1.1 NEED FOR SERVERS	1
1.2 USING THE PERSONAL COMPUTERS AS A SERVER	2
1.3 OBJECTIVE OF THE THESIS	5
1.4 ORGANIZATION OF THE THESIS	5

Chapter 2 : SERIAL AND PARALLEL I/O INTERFACE STANDARDS

2.1 INTRODUCTION	7
2.2 RS-232C SERIAL INTERFACE	8
2.2.1 Introduction	8
2.2.2 Control lines and signaling levels	8
2.2.3 Null Modem	9
2.3 DATAPRODUCTS PARALLEL INTERFACE	9
2.4 CENTRONICS PARALLEL INTERFACE	10

Chapter 3 : HARDWARE AND SOFTWARE - A BLOCK LEVEL OVERVIEW

3.1 INTRODUCTION	17
3.2 BLOCK LEVEL DESCRIPTION OF HARDWARE	19
3.2.1 Introduction	19
3.2.2 Buffer Module	19
3.2.3 Multiplexed Address/Data Module	22
3.2.4 Address Decoding Module	26
3.2.5 Main Module of the Controller	26
3.2.6 Mux./Demux. Module for Serial Interface	28
3.2.7 Line Driver and Receiver Module	29
3.3 BLOCK LEVEL DESCRIPTION OF SOFTWARE	31
3.3.1 Introduction	31
3.3.2 Central Node Software	31
3.3.3 Secondary Node Software	36

Chapter 4 : PRINT SERVER - A HARDWARE DESCRIPTION

4.1 INTRODUCTION	39
4.2 BUFFER MODULE	39
4.2.1 Brief Description of I/O Channel	39

4.2.2	Description of Address-Data Buffer	40
4.3	ADDRESS DECODING LOGIC	44
4.4	MULTIPLEXED ADDRESS/DATA MODULE	46
4.4.1	Wait State Logic	47
4.4.2	Generation of Control Signals of 8256 MUART	49
4.4.3	Address/Data Multiplexer	53
4.5	MAIN MODULE OF THE CONTROLLER	55
4.5.1	Clock Generator Circuit for Serial I/O	55
4.5.2	MUART and its associated signals	57
4.5.3	PPI and its associated signals	58
4.5.4	Interrupt Generation Logic	60
4.6	MUX./DEMUX. MODULE FOR SERIAL INTERFACE	62
4.6.1	The need for Multiplexing	62
4.6.2	Circuit Description	62
4.7	LINE DRIVER AND RECEIVER MODULE	65
4.7.1	Parallel I/O Line Drivers	65
4.7.2	Serial Line Drivers and Receivers	68
4.8	INTER-CARD BUFFER MODULE	68

Chapter 5 : PRINT SERVER - A SOFTWARE DESCRIPTION

5.1	INTRODUCTION	70
5.2	SOFTWARE FOR SECONDARY NODE	70
5.2.1	Introduction	70
5.2.2	MAIN_MODULE1 Routine	70
5.2.3	Interrupt Service Routine for INT-63H	73
5.2.4	Modem Interrupt Routine	76
5.2.5	Transmitter Interrupt Routine	81
5.2.6	Receiver Interrupt Routine	82
5.2.7	Error on Reception Interrupt Routine	83
5.3	SOFTWARE FOR CENTRAL NODE	83
5.3.1	Protocol Consideration	83
5.3.2	MAIN_MODULE Routine	87
5.3.3	Scan Interrupt Routine	87
5.3.4	Debounce Interrupt Routine	89
5.3.5	Receiver Interrupt Routine	91
5.3.6	Time-out Interrupt Routine	93
5.3.7	Transmitter Interrupt Routine	93
5.3.8	Printer Routine	98
5.3.9	Buffer Management	98
5.3.10	Check Routine	103

Chapter 6 : CONCLUSIONS AND RECOMMENDATIONS

6.1 CONCLUSIONS 105

6.2 RECOMMENDATIONS 105

6.2.1 Hardware Level 105

6.2.2 Software Level 106

107

Bibliography

**Appendix - A : Calculation of Average Worst Case Delay
for a Response from Central Node** 108

Appendix - B : Software Listing 112

Appendix - C : Circuit Layout 116

LIST OF FIGURES

Fig. 1.1	Print Server with Star Topology	4
Fig. 2.1	Null Modem	16
Fig. 2.2	Centronics Interface Signal Timing	16
Fig. 2.3	Dataproducts Interface Signal Timing	16
Fig. 3.1	Typical Star Topology	20
Fig. 3.2	Typical Interrupt Driven Centrally Controlled Bus System	20
Fig. 3.3	Basic H/W Block Diagram	21
Fig. 3.4	Block Diagram Of Buffer Module	23
Fig. 3.5	Multiplexed Address-Data Bus Module	24
Fig. 3.6	Address Decoding Module	24
Fig. 3.7	Main Module Of The Controller	27
Fig. 3.8	Mux./Demux. Module For Serial I/O	30
Fig. 3.9	Line Drivers & Receiver Module	30
Fig. 3.10	Block Diagram Of Central Node Software	33
Fig. 3.11	Block Diagram Of Secondary Node Software	37
Fig. 4.1	I/O Channel Diagram	42
Fig. 4.2	Logic Diagram Of Buffer Module	43
Fig. 4.3	Timing Diagram 1	45
Fig. 4.4	Logic Diagram Of Address Decoding Module	45
Fig. 4.5	Logic Diagram Of Wait State Circuit	48
Fig. 4.6	Timing Diagram 2	48
Fig. 4.7	ALE Generation Circuit	50
Fig. 4.8	Timing Diagram 3	50
Fig. 4.9	Generation Of Read, Write & Buffer Enable Signals	52
Fig. 4.10	Timing Diagram 4	52
Fig. 4.11	Address-Data Multiplexer	54
Fig. 4.12	Clock And MUART Chip Interconnections	56
Fig. 4.13	8255 PPI Chip Interconnection	59
Fig. 4.14	Centronics Printer Request Generation	61
Fig. 4.15	Timing Diagram 5	61
Fig. 4.16(a)	Multiplexing Logic For Serial I/O	63
Fig. 4.16(b)	Demultiplexing Logic For Serial I/O	64
Fig. 4.17	Line Drivers & Connectors For Parallel I/O	66
Fig. 4.18	Line Drivers & Receivers For Serial I/O	67
Fig. 4.19	Inter-card Buffering Circuit	69
Fig. 5.1	Flowchart Of MAIN_MODULE1 Routine	72
Fig. 5.2	Flowchart For Software Interrupt Routine (INT 63H)	74
Fig. 5.3	Flowchart Of Modem Interrupt Routine	77
Fig. 5.4	Structure Of CONTROL Packet	77
Fig. 5.5	Flowchart Of Transmitter Interrupt Routine	78
Fig. 5.6	Flowchart Of CONTROL Routine	79
Fig. 5.7	Flowchart Of SDATA Routine	80
Fig. 5.8	Flowchart Of Receiver Interrupt Routine	84
Fig. 5.9	Structure Of MESSAGE Packet	84
Fig. 5.10	Flowchart Of Error During Reception Interrupt Routine	85

Fig. 5.11	Flowchart For Scan Interrupt Routine	88
Fig. 5.12	Flowchart For Debounce Interrupt Routine	90
Fig. 5.13	Flowchart For Transmitter Interrupt Routine (Central Node)	94
Fig. 5.14	Flowchart For Timeout Interrupt Routine	95
Fig. 5.15	Flowchart For Receiver Interrupt Routine (Central Node)	95
Fig. 5.16	Flowchart For Error Updating	96
Fig. 5.17	FIFO Structure & Status	96
Fig. 5.18	Flowchart For Control Packet Reception	97
Fig. 5.19	Flowchart For Data Packet Reception	100
Fig. 5.20	Flowchart For Printer Routine	101
Fig. 5.21	Flowchart For Check Routine	104

LIST OF TABLES

Table 2.1	RS-232C Control Line Description	12
Table 2.2	DATAPRODUCTS Interface Pin Description	14
Table 2.3	CENTRONICS Interface Pin Description	15
Table 4.1	I/O Channel Description	41
Table 4.2	State Description Of Wait Generator Signal	47

ABSTRACT

The aim of this thesis is to develop a low cost PC based Print Server. This Print Server works on a Centralized Shared Server Model. The Central Node is connected to 16 Secondary Nodes through a 4-wire RS-232C serial interface.

The Central node software will be executed in background interrupt driven mode with some user defined task running in fore-ground. At the Secondary nodes also, the file transmission to central node for printing will be done in background interrupt driven mode to minimize the CPU time used up for this task. Thus all nodes are efficiently used as far as possible.

The Central Node supports CENTRONICS and DATAPRODUCTS Parallel Interface Printers as well as the RS-232C based Serial Interface Printer.

CHAPTER 1

INTRODUCTION

1.1 NEED FOR SERVERS:

A recent trend in Computer Systems is to distribute computation among several physical processors. There are two schemes of handling the above situation , multiprocessor systems and distributed systems. [1]

In the distributed system, processors may vary in size and function, including from small microprocessors, workstations, minicomputers and large general purpose computer systems. There are four major reasons for building distributed systems : resource sharing, computation speed-up, reliability and communication. Here, in this project, we are more interested in resource sharing.

If a number of different sites with different capabilities are connected to each other, then a user at one site may be able to use the resource available at another site. For example, a user at site-A may be using a Laser printer that is provided at site-B. In general, resource sharing in a distributed system provides mechanisms for sharing files at remote sites, processing information in a distributed database, printing files at remote sites, using remote specialized hardware devices, such as high-speed array processor, and other operations.

There are various means by which access to the shared resources is accomplished. They are data migration, computation migration or job migration. If for instance, the computer becomes saturated with too much work at one site, the work can be loaded through a distributed system onto another computer in the network. Such load-sharing permits a more even and better utilization of resources.

1.2 USING THE PERSONAL COMPUTER AS A SERVER:

The concept of a server implies the use of one or more personal computers to perform specific tasks for a number of other PCs. The most common functions are disk, file and print servers. Some vendors also classify a server as one that provides an gateway to other networks.[7]

A disk server provides low-level support, typically the basic read and write operations to disk sectors using the disk controller commands. In contrast, a file server is a higher level support mechanism, performing such functions as lock-out and dynamic allocation of space on the disk. A print server is a support mechanism by which that those PCs which do not have a printer connected to them directly, but can still use a print facility by just being attached to the system.

The PC user have two primary options when selecting a server:

- a) a dedicated versus a shared server.
- b) a centralized or a distributed server.

The dedicated server has better performance than a shared server, since it performs only to one or a few specific tasks. A shared server's performance is almost always poorer because of the continuous interrupts made to its operating system to provide service to multiple users. In addition, a shared server shares its disk with multiple users, which decreases the actual capacity available to the community. A dedicated server is usually more secure than a shared server, but it is more expensive. If traffic is not high, shared server should be used.

The selection of server also entails the choice of a centralized or a distributed server. A centralized server means that one PC is used to provide the service. Distributed servers entail the use of more than one PC for the support function.

A distributed server system can be very complex. Servers with a large databases, multiple copies of data, and short response time on updates require a complex distributed environment. If possible, complexities should be avoided because of increased risk of system failures, data inconsistencies, and increased costs.

One way of implementing centralized server on PC is by using the Star Topology as shown in Fig.1.1. It is easy to control, the software is not complex and the traffic flow is simple. All traffic emanates from the hub of the site, the central site. The central site is responsible for routing traffic through it to the other components. The central site

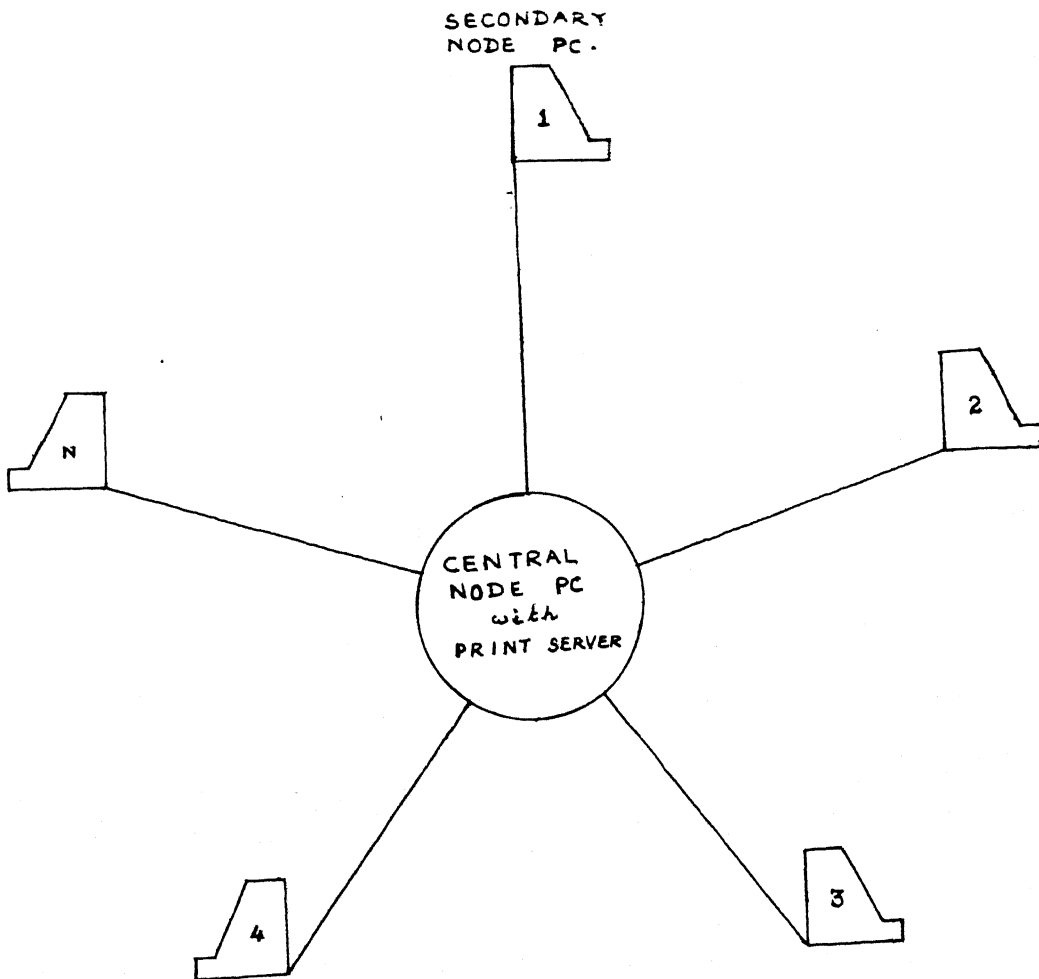


FIG. 1.1: PRINT SERVER WITH STAR TOPOLOGY

is also responsible for fault isolation. Fault isolation is relatively simple in a Star Network because the lines can be isolated to identify the problem. However, the star topology is subjected to potential bottleneck and failure problems at the central site.

1.3 OBJECTIVE OF THE THE THESIS:

The aim of this thesis is to develop a print server for Personal Computer environment. The chosen configuration is a centralized server using a star topology. In a Lab. environment, where the number of the users are large and a large number of PCs are available, it has been seen that the providing every user PCs with a separate printers attached to them is a very tall order. The cost of a printer is comparatively high, so each user PCs could be connected in a Star Topology as shown in Fig.1.1 to central node using COM-port of the PC without any extra hardware needed. It has been designed using a Multifunction Universal Asynchronous Receiver Transmitter chip (8256) at the central node, while at the secondary node the communication is through COM-port of PC. The software for print server is written using a shared server approach, thus utilizing the resource in an optimum way.

1.4 ORGANIZATION OF THE THESIS:

Chapter-2 summarizes the salient points of various interfaces implemented in this thesis like RS-232C Serial Interface, DATAPRODUCTS and CENTRONICS Parallel Interfaces for line printers.

Chapter-3 presents the block diagram level overview of how the hardware and software is implemented in this thesis. It discusses each sub-modules in great details.

Chapter-4 presents the detail description about the hardware implemented. It also contains the various timing constraints on the handshake signals. The chip level description of the circuit is given to facilitate the full understanding of it.

Chapter-5 discusses the flowcharts and details of the software developed in this thesis. Each of the major routines are described in an elaborate manner.

Chapter-6 gives the conclusions and suggestions made for some additional features which can be incorporated in the Print Server.

Appendix-A gives the theoretical analysis of the average worst case delay to the request from the user by the central node. Appendix-B includes the listing of the application program written in 'C' language for converting the wordstar document files into file which can be transferred from the secondary node to the central node. The remaining software of the Print Server written in 8086/88 Assembly language is given in floppy disk. Appendix-C gives the circuit diagram and the layout of the cards developed for the Print Server.

CHAPTER 2

SERIAL AND PARALLEL I/O INTERFACE STANDARDS

2.1 INTRODUCTION:

There are many different types of peripheral devices. Many of them are electro-mechanical devices and hence they transfer data at a slower rate. One method of transferring information to or from a computer is through a serial link, i.e. one bit at a time over a single pair of conducting wire with each bit occupying an interval of time having a specified length. The serial communication can either be through asynchronous means where special bit pattern separate the characters or synchronous means where special "sync" character are used during idle time. One of the most popular asynchronous standard is RS-232C Interface which is discussed in Section-2.2.

The other mode of transferring data is through a parallel link. It improves the speed of transfer but at a higher cost. The Line Printers work at reasonably good speed [100 character per second (cps) to 600 line per minute (lpm)], and use parallel i/o. Two of the standardized Parallel Interface for Line Printers are DATAPRODUCTS and CENTRONICS Interfaces. These interfaces have been implemented in this thesis and are discussed in sections 2.3 and 2.4 respectively.

2.2 RS-232C SERIAL INTERFACE:

2.2.1 Introduction:

EIA RS-232C is essentially same as the CCITT V.24 standard. It is concerned with sending serial bit stream between interfaces or terminals and communication equipment.

Modulator/demodulator circuit called modems are placed at the ends of communication links, so as to drive the signals over long distances, not feasible by direct connection of the lines.

2.2.2 Control Lines and Signaling Levels:

RS-232C standard are the definitions and symbolic representations of control lines that run between an interface or terminal and its modem. The definitions are summarized in Table-2.1 [3]. The connections are most often made using 25-pin D-Shell Connectors.

Only eight lines are normally needed while using with modems over a direct telephone. For transmission, a Request To Send (RTS) is sent to the modem, which acknowledges by Clear To Send (CTS). The transmission begins over Transmit Data (TxD) line. For reception, the Received Line Signal Detect (RLSD) line is activated by modem at the other end of transmission link. The reception begins over Receive Data (RxD) line. The Data Set Ready (DSR) line indicates that the modem is turned on and is in its data mode. This line has to be on while transmitting or receiving. On a switched system, signals are sent to modem over Data Terminal Ready (DTR) line to control the switching of modem so that a link is

established. The Ring Indicator (RI) line is activated by the modem to indicate to the interface that modem is receiving a ringing signal.

There RS-232 Line Drivers to convert TTL level signals to RS-232 levels, similarly RS-232 Line Receiver will do vice-versa. A TTL level "1" signal is converted into -12V (ranging from -3V to -25V) and TTL level "0" signal is converted into +12V (ranging from +3V to +25V). The range from -3V to +3V is considered as an indeterminate region. The TxD and RxD line will be in Marking state (logical level=1) during idle time and transmission starts with one bit going to Space state (logical level=0).

2.2.3 Null Modem:

The term 'Null Modem' implies absence of a Modem. Most RS-232 equipment are expected to be attached to telephone system using a Modem. If a computer is connected to printer which located close-by, then there is no need of Modems.

The null modem cable reverses, or swaps connections. The transmitted data (TxD) line of one is connected to received data (RxD) line of other interface and vice-versa. CTS of one interface is swapped with RTS of another interface. DSR and DTR are similarly swapped. The null modem is shown in Fig.2.1.

2.3 DATAPRODUCTS PARALLEL INTERFACE:

This interface is similar to CENTRONICS Interface, an 8-bits per byte parallel TTL compatible interface. The interface pin definitions are described in Table-2.2 [8].

In this interface, the data which is put out by the host is latched on the leading edge of a STROBE signal. Once the STROBE signal becomes high, the DATA REQUEST line is lowered by the printer. It means that printer is busy with the data and no new data can be accepted. The STROBE line is then automatically lowered. Once the data is processed by the printer, the DATA REQUEST line is raised high, this normally triggers an interrupt. So the host will again output a byte and latch it on rising edge of STROBE. The interface signal timings are shown in the Fig.2.3.

The status information of the printer like READY and ON-LINE are also made available. If READY and ON-LINE are both low, then the data put out by the host is lost. Hence on power-on initialization, READY and ON-LINE is made high, which after some time makes DATA REQUEST line low and the whole process can begin.

2.4 CENTRONICS PARALLEL INTERFACE:

This interface is an 8-bits per byte parallel TTL compatible interface. It uses 8-bit ASCII superset data transfer code without parity check and is connectable via an Amphenol 57-40360-12-D56 female connector to most of the hosts using this interface. The interface pin assignments and definitions are described in Table-2.3 [10].

The interface signal timing is shown in Fig.2.2. Each character transmitted to the printer must be clocked by a -STROBE signal. When -STROBE is received, it sets the BUSY signal high to inform the host that the printer is no longer

able to receive further data. When the character has been processed, an -ACK is sent and BUSY is made low to inform the host that further data can be sent.

When the printer is powered on, a high level BUSY signal is output through the interface until the printer is put in the ready state to accept data. At the end of the initialization an -ACK pulse is sent to the host before lowering the BUSY signal.

In the normal mode of interface, either BUSY and STROBE or ACK and STROBE are used for two-wire handshake control signals. In this project, ACK and STROBE are used for handshaking. Few other lines as described in Table-2.3 provides the printer with status information.

TABLE 2.1 : RS-232C CONTROL LINE DEFINITIONS [3]

SYMBOL EIA [CCITT]	PIN	NAME	DESCRIPTION
AA [101]	1	Chassis Ground	Equipment Ground
AB [102]	7	Signal Ground	Common ground for all signals.
BA [103]	2	Transmit Data	Output data to modem.
BB [104]	3	Receive Data	Input data from modem.
CA [105]	4	Request To Send	To turn modem's transmitter ON and OFF during half duplex.
CB [106]	5	Clear To Send	Modem indicates it is ready to transmit.
CC [107]	6	Data Set Ready	Modem indicates it is not in test mode.
CF [109]	8	Rec. Line Signal Det.	Modem indicates it is receiving signal from other end modem.
CD [108/2]	20	Data Terminal Ready	Prepares modem to be connected to communication link.
CE [125]	22	Ring Indicator	Modem indicates ringing signals detected on link.
CG [110]	21	Signal Qual. Det.	Modem activates when low probability of error in rec. data.
CH [111]	23	Data Rate Select	Indicates to modem one of two synchronous data rates.[DTE]
CI [112]	25	..	Modem indicates to interface 1 of 2 syn. data rates.[DCE]
DA [113]	24	Trans. Sig. Elm. Timing	Provides modem transmitter with signal timing.[DTE]
DB [114]	15	..	Modem provides interface with transmitter signal timing.[DCE]
DD [115]	17	Rece. Sig. Elm. Timing	Modem provides interface with signal timing.

TABLE 2.1 [Continued]

SYMBOL	PIN	NAME	DESCRIPTION
EIA [CCITT]			
SBA[118]	14	Secondary	To modem for outputting low Trans. Data rate data.
SBB[119]	16	Secondary	From modem for inputting low Rece. Data rate data.
SCA[120]	19	Secondary	To modem to turn on modem's Req.To Send secondary transmitter.
SCB[121]	13	Secondary	Modem indicates secondary Clr.To Send transmitter is ready.
SCF[122]	12	Sec. Rec.	Modem indicates signal is Line.Sig.Det. detected on sec. link.

TABLE 2.2 : DATAPRODUCTS INTERFACE PIN DESCRIPTION

SIGNAL	CONNECTOR Sig.(Ret.)	DESCRIPTION
DATA REQ.	E (C)	Sent by Printer to synchronize data transmission. Remains true until DATA STB. is recv., goes low by 100nS
DATA STB	j (m)	Host indicates to Printer to accept information on data lines. Should remain true till DATA REQ. goes low.
DATA-1	B (D)	Data-1 is least significant bit
DATA-2	F (J)	while Data-8 is most significant bit
DATA-3	L (N)	
DATA-4	R (T)	
DATA-5	V (X)	
DATA-6	Z (b)	
DATA-7	n (k)	
DATA-8	h (e)	
VFU CONT.	p (s)	Optional control from host. Used for VFU Control.
READY	CC(EE)	Printer sends it when no error condition exists.
ON-LINE	y (AA)	Printer indicates it when operator has activated ON-LINE while READY is active. Enables interface activity.
INTERFACE VERIFY	x to v	Jumper in Printer connector. Continuity informs host that conn. is o.k.
+5V DC	HH	Supply voltage for exerciser only.

TABLE 2.3 : CENTRONICS INTERFACE PIN DESCRIPTION

NAME	PIN		SOURCE	DESCRIPTION
	Sig.	Ret.		
-STROBE	1	19	Host	Signal controlling data transmission, lasts for min. of 1uS.
DATA-1	2	20	,,	Data Bit-1 is the least significant bit while Data Bit-8 is the most significant bit. The data signal must be stable from 1uS before STROBE goes low to 1uS after STROBE goes high.
DATA-2	3	21		
DATA-3	4	22		
DATA-4	5	23		
DATA-5	6	24		
DATA-6	7	25		
DATA-7	8	26		
DATA-8	9	27		
-ACK	10	28	Printer	Indicates data has been recv. It lasts for 2.5uS to 6.6uS.
BUSY	11	29	Printer	Indicates printer not ready to receive data.
PE	12	14	,,	Indicates printer has run out of paper.
SLCT	13	-	,,	Indicates READY has been selected and printing can occur.
-PRIME	31	30	Host	Not used.
-FAULT	32	-	Printer	Indicates printer is OFF-LINE.
-SLCT IN	36	-	Host	Hardware grounded in Printer.
+5V DC	18	-	Printer	Can source upto 100mA current.
Chs. GND	17	-	-	Frame ground.
Sig. GND	- 14,16,33		-	Common ground signal

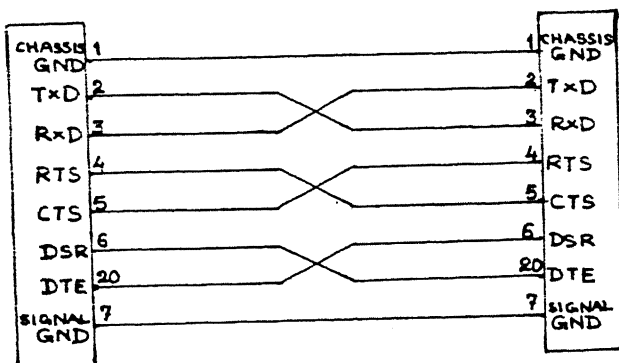


FIG. 2.1 : NULL MODEM

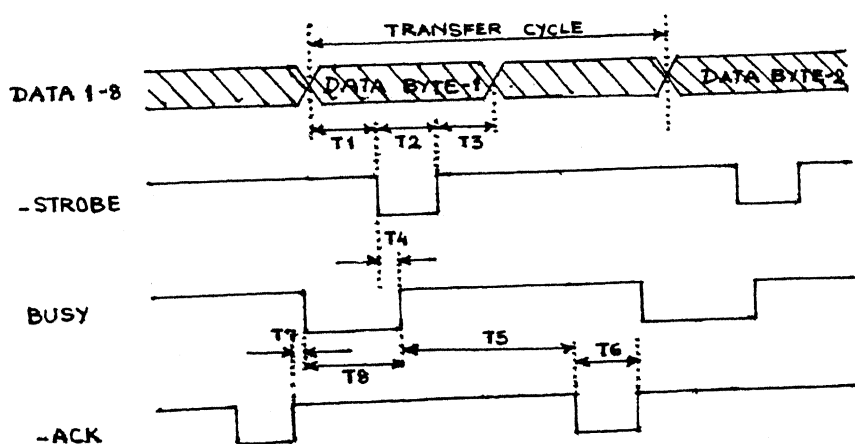


FIG. 2.2 : CENTRONICS INTERFACE SIGNAL TIMING

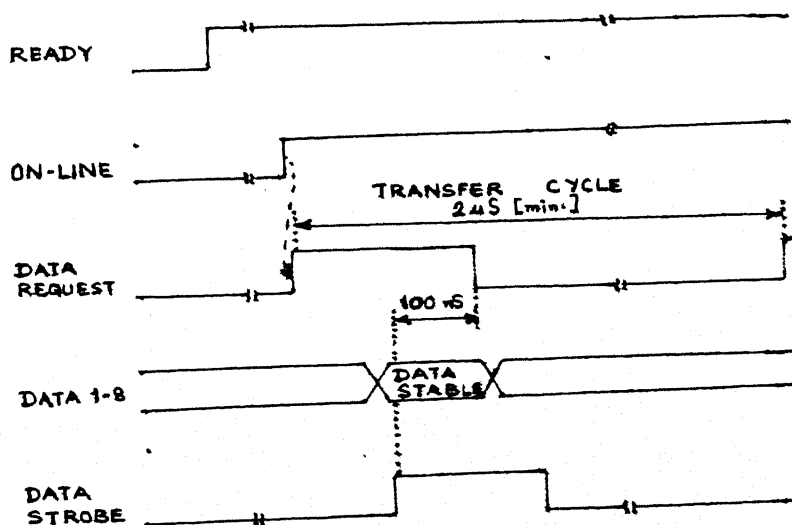


FIG. 2.3 : DATAPRODUCTS INTERFACE SIGNAL TIMING

CHAPTER 3

HARDWARE AND SOFTWARE : A BLOCK LEVEL OVERVIEW

3.1 INTRODUCTION :

In a microcomputer system based on a Star Configuration, one microcomputer acts as a central node (master), having separate lines connected to all other microcomputers (slaves) as shown in Fig.3.1. Typically, when node-A wants to send a message to node-B, a Request To Send (RTS) message is sent to the switch-S (master) which in turn will establish a path to node-B only on receiving a Clear To Send (CTS) message from node-B. Many star-technology based systems [6] operate in a two-level hierarchical model, where the central switch works as a message switch, as well as a general-purpose data processing facility. The interface between the master and slave nodes is over a 20-mA current loop serial or RS-232 serial interface, asynchronous or synchronous bit line. The central switch function is more complex in a star configuration than in a bus-based system since the controller has to control a larger number of message paths concurrently.

The Interrupt Driven Centrally Controlled Bus has separate control lines as shown in Fig.3.2. The controller receives random requests from nodes which are ready to transmit data and queues them up. In this system, a

Request To Send (RTS) message is sent spontaneously by the sending node to the controller which then sends an acknowledgment to the requesting node. If no acknowledgment is received after a random delay, the sending node re-transmits the request.

In a Slotted Centrally Controlled Bus system, time is divided into intervals called slots. The transmitting node sends a message in its assigned time-slots. The controller assigns slots uniquely to nodes based on a 2^N evenly spaced slots in a given time frame. The service request from a node is made in the slot dedicated to it.

The hardware and software implementation for the Print Server described in this thesis implements an integrated approach wherein it incorporates many of the characteristics of the below three methods, i.e.

- a) Star Topology
- b) Interrupt Driven Centrally Controlled Bus
- c) Slotted Centrally Controlled Bus

The central node and the secondary node are physically connected in a star topology using RS-232 serial link interface. This interface uses 2-wire handshake with Duplex communication facility. The Request To Send (RTS) message is sent by secondary node using separate control lines of central node to interrupt it when needed. The central node uses MUART's (8256) timer to divide the time into various slots. The Request To Send (RTS) message or signal from the secondary node is accepted only during its assigned slot.

The central node sends an acknowledgment to the requesting node and allows the link to be used by the requesting secondary node to download the whole file which has to be printed on the printer attached to the central node.

The central node achieves data flow control using Clear To Send (CTS) signal. The central node will make the CTS line inactive to inhibit the data from being sent by the secondary node when its buffer can overflow while making CTS line active when it wants to allow the data to be obtained from the secondary node. The buffer management at central node is done using pointers and flags with circular buffer.

3.2 BLOCK LEVEL DESCRIPTION OF HARDWARE:

3.2.1 Introduction:

As shown in Fig.3.3, the block diagram of the Print Server Controller is made up of the following six sub-blocks:

- a) Buffer module for address, data and control bus.
- b) Multiplexed address/data module for 8256.
- c) Decoder module for 8255 and 8256.
- d) Main module of the controller.
- e) Mux./Demux. module for serial interface.
- f) Line driver and receiver module.

The brief description of each module is given in this section.

3.2.2 Buffer Module:

The buffered signals available on I/O channel of PC can drive only two Low-power Schottky (LS) TTL loads per I/O

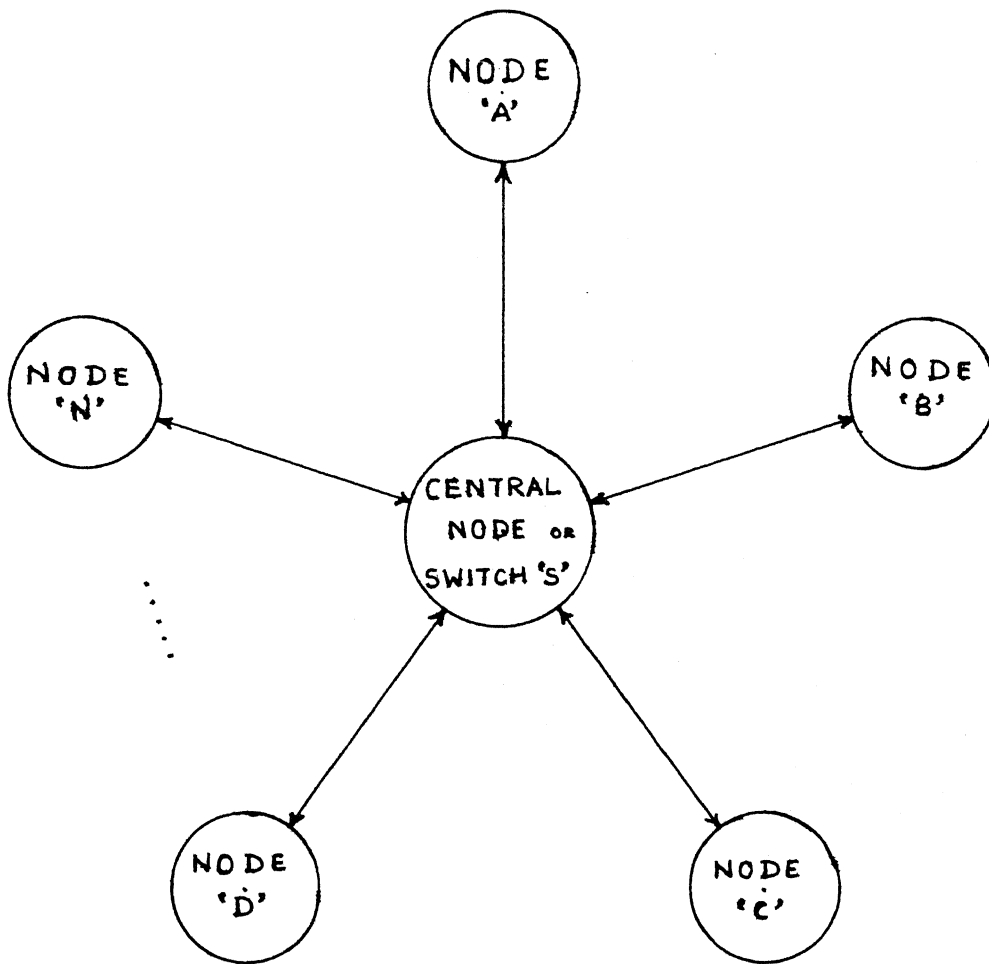


FIG. 3.1 : TYPICAL STAR TOPOLOGY

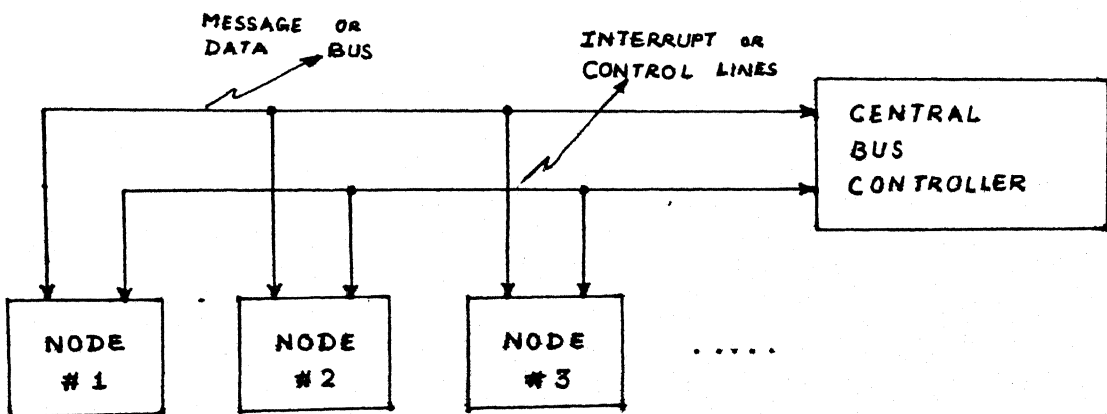
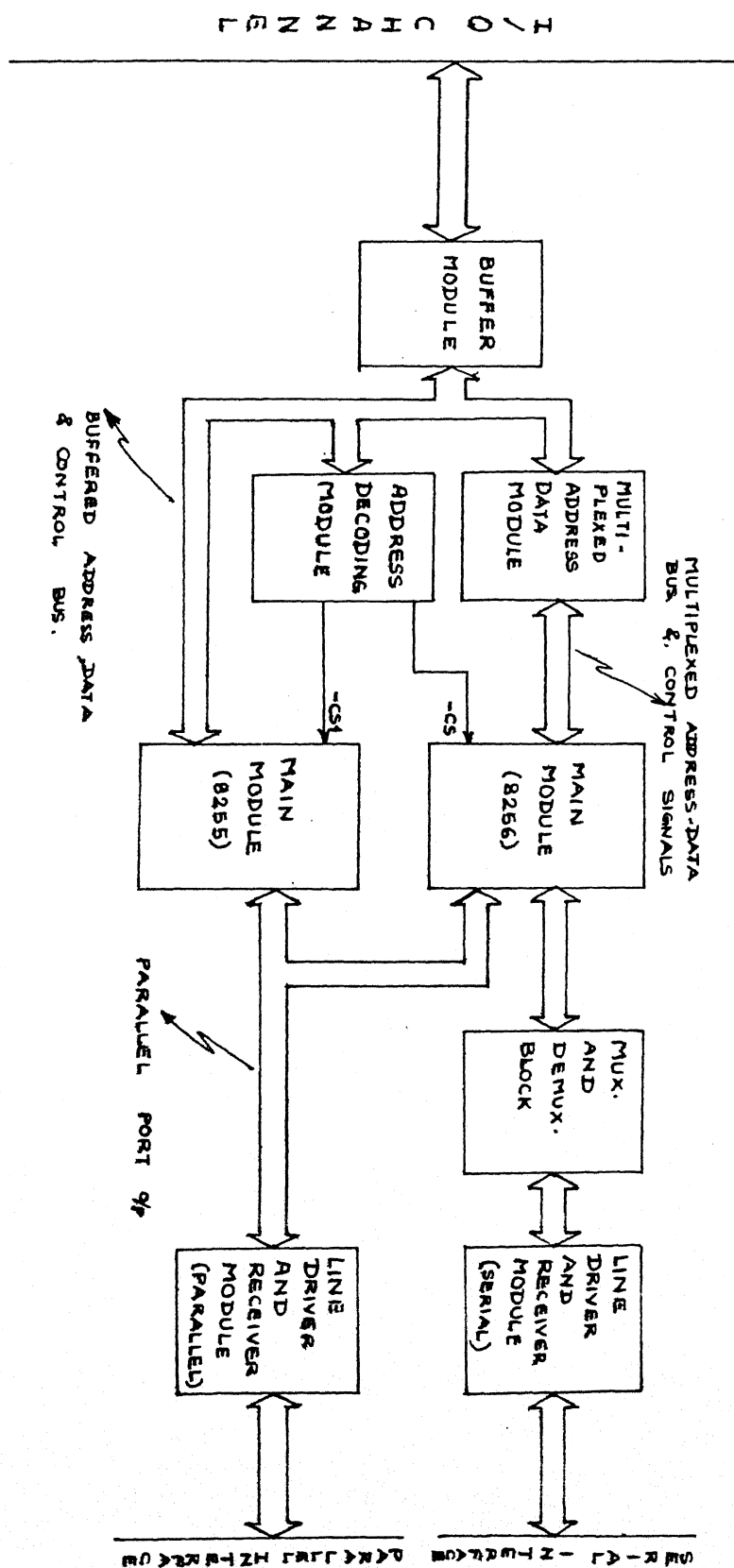


FIG. 3.2 : TYPICAL INTERRUPT DRIVEN CENTRALLY CONTROLLED BUS SYSTEM

FIG. 3.3 : BASIC H/W BLOCK DIAGRAM



channel slot, hence it is a standard practice to buffer the signals which are used in the card. The address bus (only A0 to A9 are needed for partial i/o decoding), data bus (D0 to D7) and various signals from control bus are buffered. The buffered control signals are I/O READ (-IOR), I/O WRITE (-IOW), System Clock (CLK), Address Latch Enable (ALE), System Reset (RESET) and Address Enable (AEN). Data bus is buffered using bi-directional buffers (74LS245), while Address and Control buses are buffered using unidirectional buffers (74LS244). The block diagram of Buffer module is shown in Fig.3.4. The unidirectional buffer is enabled permanently, while the bi-directional buffer is enabled by a signal generated using chip select, -IOR and -IOW. The detailed discussion is given in Chapter-4. The direction of bi-directional buffer is controlled by -IOR signal. Thus in the inactive state, the buffer is in the write mode for the processor and contention on data bus is avoided.

3.2.3 Multiplexed Address/Data Module:

The INTEL 8256 Multifunction Universal Asynchronous Receiver Transmitter (MUART) is designed to be used for serial asynchronous communication while providing hardware support for parallel I/O, timers and interrupt control. Its versatile design allows it to be directly connected to iapx-86 or iAPX-88. It requires multiplexed Address-Data bus from 8086/88 processor and ALE signal for address latching. In the I/O channel the address and data bus are de-multiplexed

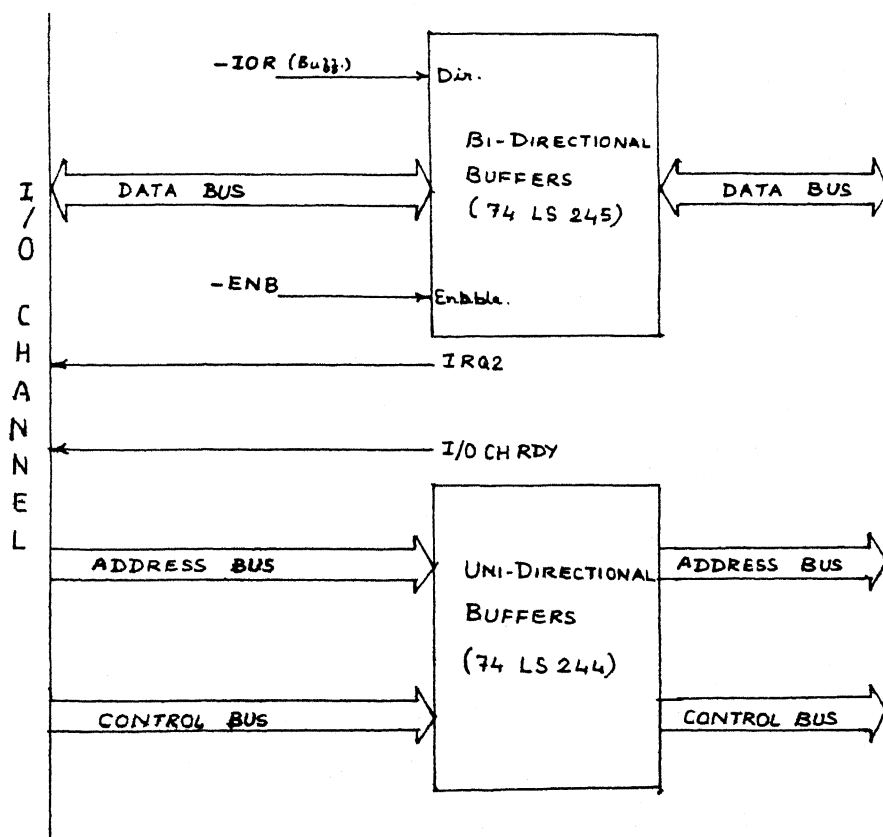


FIG. 3.4 : BLOCK DIAGRAM OF BUFFER MODULE

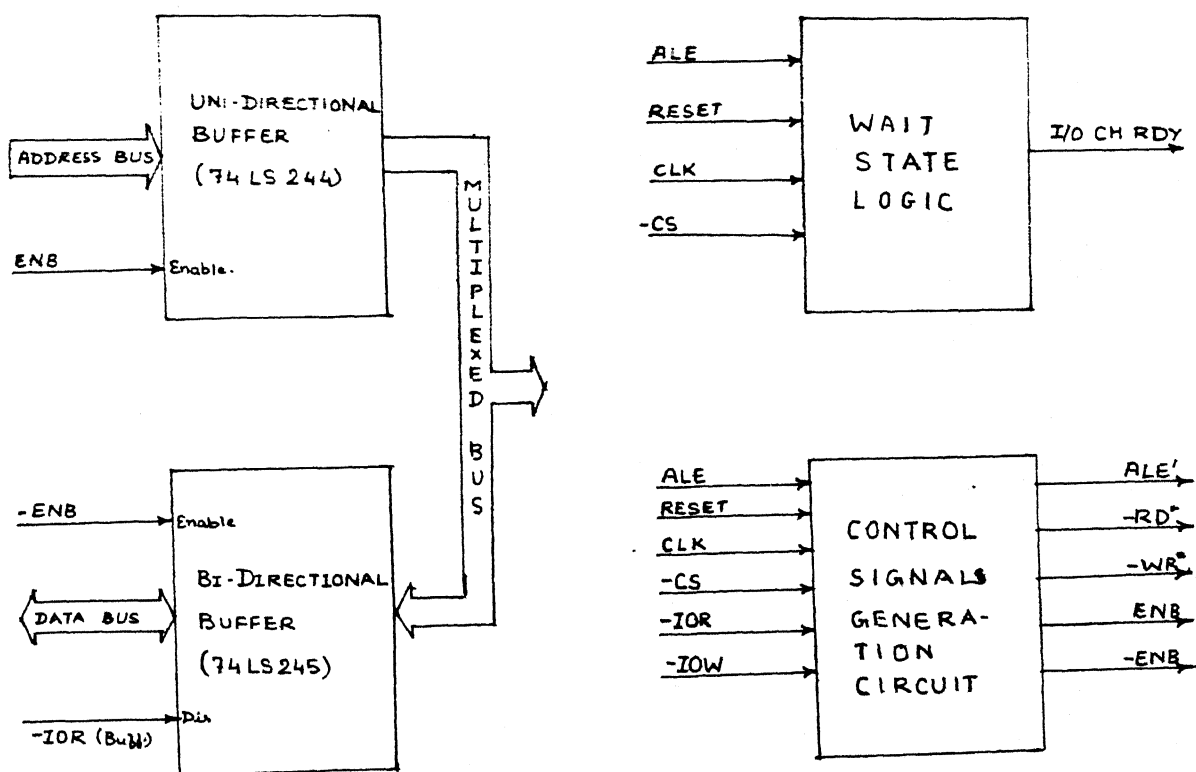


FIG-3-5 : MULTIPLEXED ADDRESS- DATA BUS MODULE

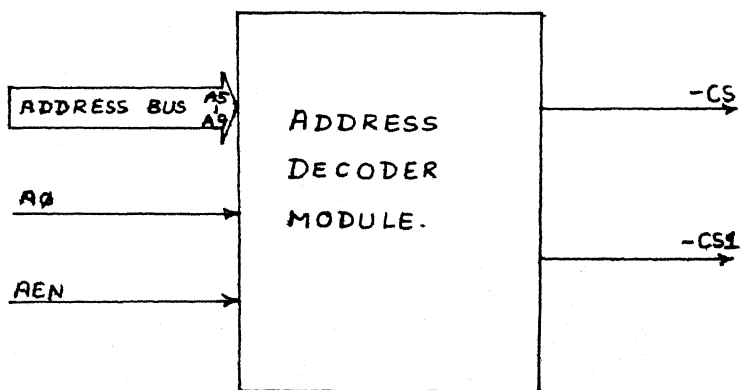


FIG- 3.6 : ADDRESS DECODING MODULE

as required by other peripheral chips. To resolve the above problem the address and data buses are re-multiplexed, the -IOR and -IOW signals are delayed properly and ALE signal is regenerated. The block level diagram of Multiplexed module is shown in Fig.3.5.

The card inserts two wait states between clock states T3 and T4 of processor machine cycle. The system board of PC inserts one wait state for any I/O operation not on system board, hence the wait state logic of the card pulls the I/O Channel Ready (I/O CH RDY) line of I/O channel low for two T-states and in effect increasing one more wait state. As a result of it -IOR and -IOW signals are elongated. The ALE signal used by system board to de-multiplex the address-data bus is again needed by 8256 at slightly later time for address latching of the re-multiplexed address-data bus. As a result ALE has to be freshly regenerated to meet 8256 specification. The -IOR and -IOW signal are delayed by one T-state by using a Latch circuitry. The 8256 Read (-RD) and 8256 Write (-WR) is generated using -IOR and -IOW and their delayed version. The chip select, -IOR and -IOW are used to generate Buffer Enable (X and -X) signals. The X and -X signals are complementary signals and are used to switch either address buffer or data buffer, thus when address buffer is open the data buffer will remain closed and bus contention at 8256 end is avoided but simultaneously generating multiplexed address-data bus.

3.2.4 Address Decoding Module:

The 8256 MUART has 16 addressable registers and 8255 chip has 4 addressable registers. When 8256 is connected to INTEL 8086/88 system the chip select has to be qualified by address bit A0 = 0, thus it requires all 16 I/O address at even locations. The prototype card slot on which this card is designed provides 32 contiguous I/O locations. The 8256 uses all the even numbered I/O locations in the prototype card, hence 8255 has to be configured at 4 odd numbered I/O locations. A partial address decoding is done in the card as recommended in IBM PC/XT Technical Reference Manual [9]. The address decoding is qualified by AEN signal to prevent the card from responding when DMA floats the same I/O address. The block diagram of address decoding module is shown in Fig.3.6. The Schottky version of decoder chip (74S138) is used to make the propagation delay minimum.

3.2.5 Main module of the controller:

The MUART 8256 chip is used for serial asynchronous communication, for timing of events, for parallel I/O and for interrupt control of various events. The 8255 Parallel Peripheral Interface (PPI) chip is used for parallel I/O.

The serial asynchronous communication and timer operation of 8256 needs a clock to feed the five timers and the baud rate generator. This clock has to be 1,2,3 or 5 multiples of the internal clocking frequency (1.024 MHz), so a crystal of 6.144 MHz is used alongwith a clock generator chip (8284-A). The clock generator gives a clock output of

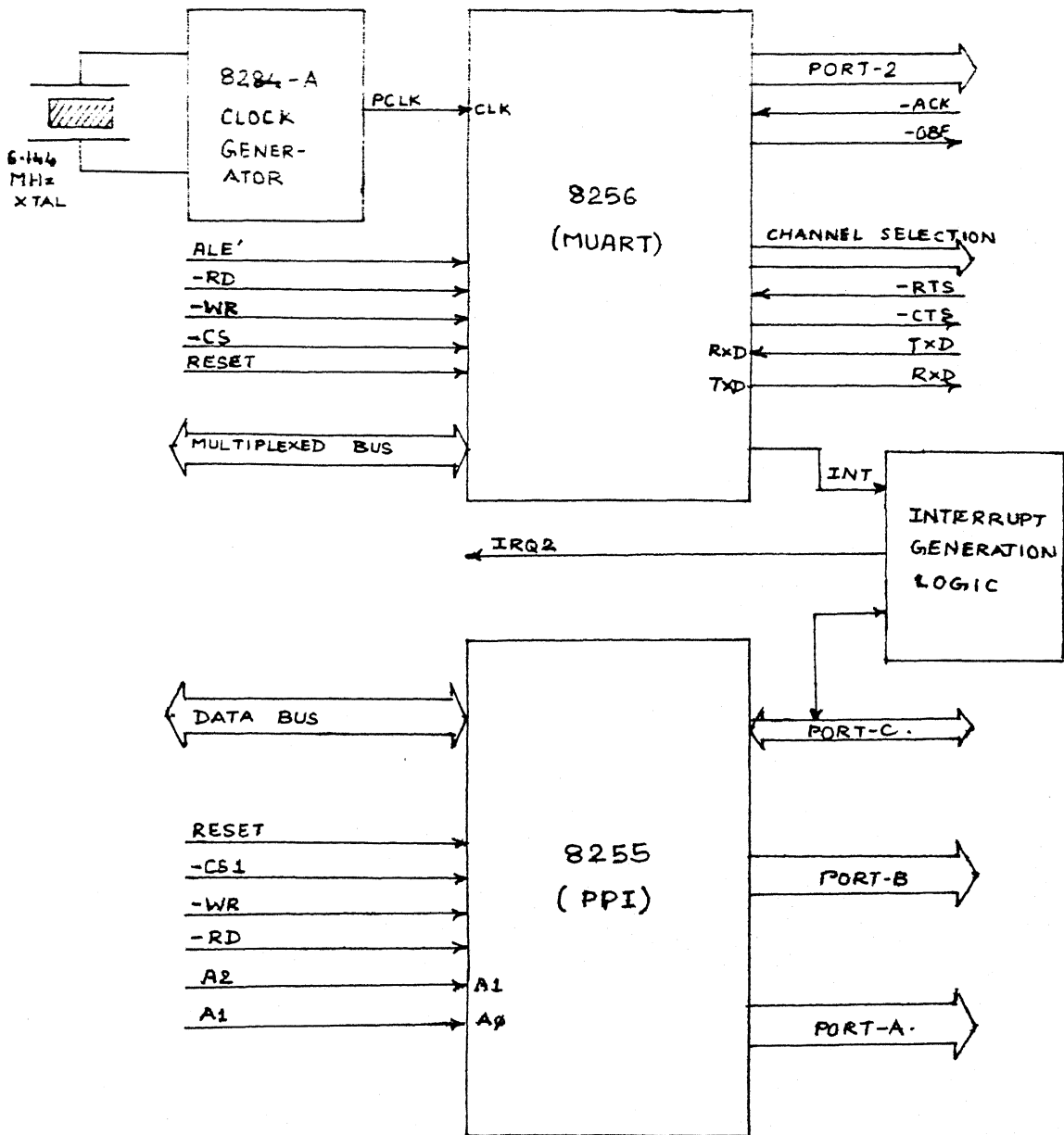


FIG. 3.7: MAIN MODULE OF THE CONTROLLER.

1.024 MHz which is equal to the internal clocking frequency. The serial communication lines (TxD and RxD) of 8256 is used alongwith two port pins of 8256 acting as Request To Send (-RTS) and Clear To Send (-CTS) to accomplish the 2-wire handshake protocol for serial I/O communication. The re-multiplexed address-data bus is connected to 8256 address-data lines.

The parallel port-2 of 8256 is used as an output handshake mode with port-1 pins working as a handshake lines. The port-2 is used for DATAPRODUCTS Parallel Interface Line Printer. The port-1 is bit programmable I/O port with 2 lines as input and remaining 6 lines as output pins. The 4 lines amongst the 6 output lines is used for serial channel selection and 1 line working as -CTS line.

The 8255 PPI has three 8-bit ports : Port-A and Port-B is used in simple output mode without handshake signals for two CENTRONICS Interface Line Printers. The Port-C is used as control and status port. The CENTRONICS Interface requires STROBE pulse and generates ACKNOWLEDGE pulse. The acknowledge pulse is latched and its level is stored on one of the Port-C pin and clearing of the latch is done by pulsing one of the other Port-C pin.

3.2.6 Mux./Demux. Module for serial interface:

The 8256 MUART support only one serial I/O communication channel. Here it is required to serve 16 serial I/O channel using only 8256 MUART, so multiplexing the signals from the 16 channel is required. The 16

secondary node will communicate with the central controller on their TxD lines, as a result 16 TxD lines of secondary node is multiplexed using 16 to 1 Multiplexer and the output is fed to RxD line of 8256 MUART. Similarly 16 secondary node send the request for transmission on -RTS lines, this is again multiplexed by 16 to 1 Mux. and fed to one of the input pin of Port-1.

The 8256 MUART transmits data to secondary node on its TxD line. It has to communicate with any one of the 16 secondary nodes on their RxD lines. Hence the TxD line of 8256 MUART is fed to a 1 to 16 De-multiplexer which then is connected to 16 RxD lines of secondary nodes. Similarly 8256 MUART sends the clear to transmit signal to secondary node on one of the output pin of Port-1 which is de-multiplexed by 1 to 16 Demux. and sent on -CTS lines of secondary nodes. The address selection of one of the serial channel is done through 4 lines of Port-1. The block level diagram of the Mux./Demux. module is given in Fig.3.8.

3.2.7 Line Driver and Receiver Module:

The secondary node and central node communicate through a cable supporting RS-232C standard. The inputs to central node from secondary node are at RS-232 levels while the card works on TTL levels, so Line Receiver [1489] chips are used to do the level translation. The outputs from central node to secondary nodes are at TTL levels which have to be converted to RS-232 level using Line Driver [1488] chips.

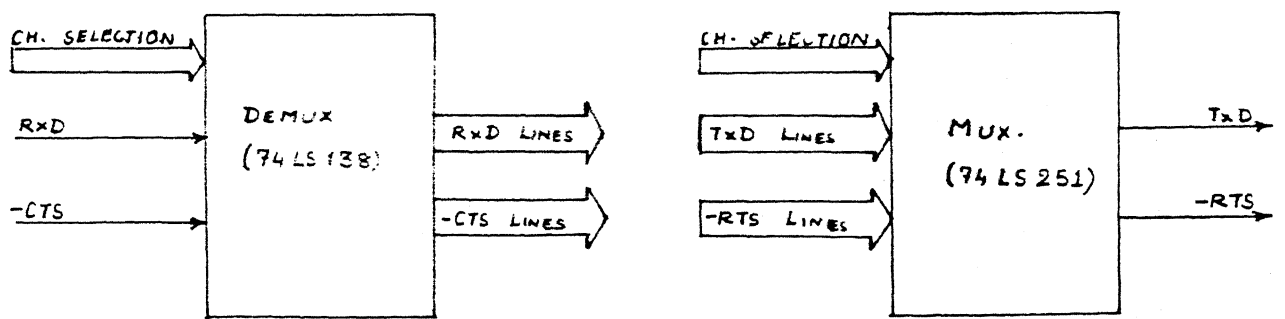


FIG. 3.8 : Mux./DEMux. MODULE FOR SERIAL I/O.

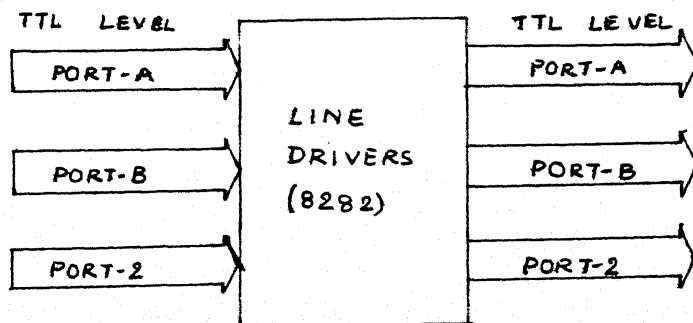
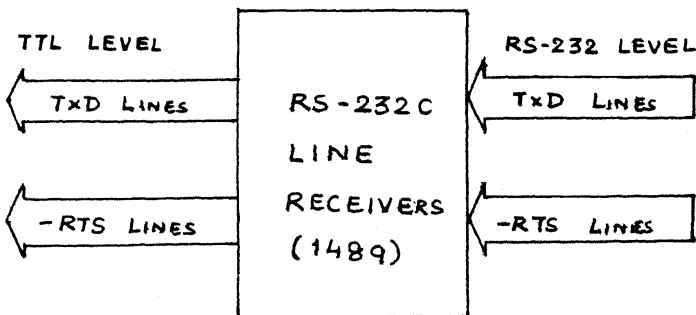
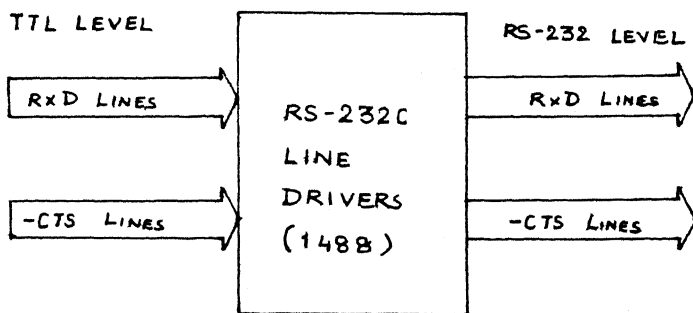


FIG. 3.9 : LINE DRIVERS & RECEIVER MODULE.

The parallel ports of 8256 MUART and 8255 PPI are connected to Line Printers by a flat ribbon cable. The parallel port cannot drive such a long cable, hence the signals are buffered by TTL Line Driver [8282] chips. The block level diagram of the this module is shown in Fig.3.9.

3.3 BLOCK LEVEL DESCRIPTION OF SOFTWARE:

3.3.1 Introduction:

The software required at central node and at secondary nodes are of different dimensions, hence they are developed in different ways and so described in different sub-sections. The server is based on centralized Shared Server model. The program is loaded into the memory at the time of system initialization and made resident in memory.

3.3.2 Central Node Software:

The block diagram of the central node software is shown in Fig.3.10. The software is made up of two separate modules each containing several procedures. The modules are identified by dotted box while the procedures are identified by solid box. Two or more procedures connected by a solid line means the procedure above calls the below procedure. The procedures without any solid lines connected are interrupt procedures. They are executed only when interrupt occurs on IRQ-2 line of I/O channel.

The Print Server uses normal priority of 8256 MUART and uses polling schemes for 8255 PPI requests. The priority of the interrupt procedures are:

- a) Debounce Timer Highest
- b) Receiver Time-out
- c) Receiver Interrupt
- d) Transmitter Interrupt
- e) Scan Timer
- f) Line Printer Interrupt Lowest

The I/o channel doesn't provide for cascading of slave interrupt controller and since only one Interrupt Request (IRQ-2) line is available we program the 8256 MUART in Normal Mode instead of Nested Mode. At any given time only one interrupt source for 8256 MUART exists, so there is no problem of starving out any lower priority interrupts. Whenever the interrupts of 8256 MUART are invoked the request from the CENTRONICS Line Printer is checked by reading the Port-C of 8255. Thus printers are not really working on interrupt mode. During idle time Scan Timer interrupt occurs every 10mS and during busy time Receiver Interrupt will occur whenever bytes are received, thus printer requests are met accordingly without much degradation in printing speed.

On power-up or during software reset, AUTOEXEC.BAT batch file is executed. In this batch file the MAIN_MODULE is also executed. The MAIN_MODULE checks for a global flag stored at a fixed memory location. If the flag is found then the resident portion of the program is not re-loaded in the memory to prevent memory wastage due to multiple copies of the program, but if the flag is not found then it sets the

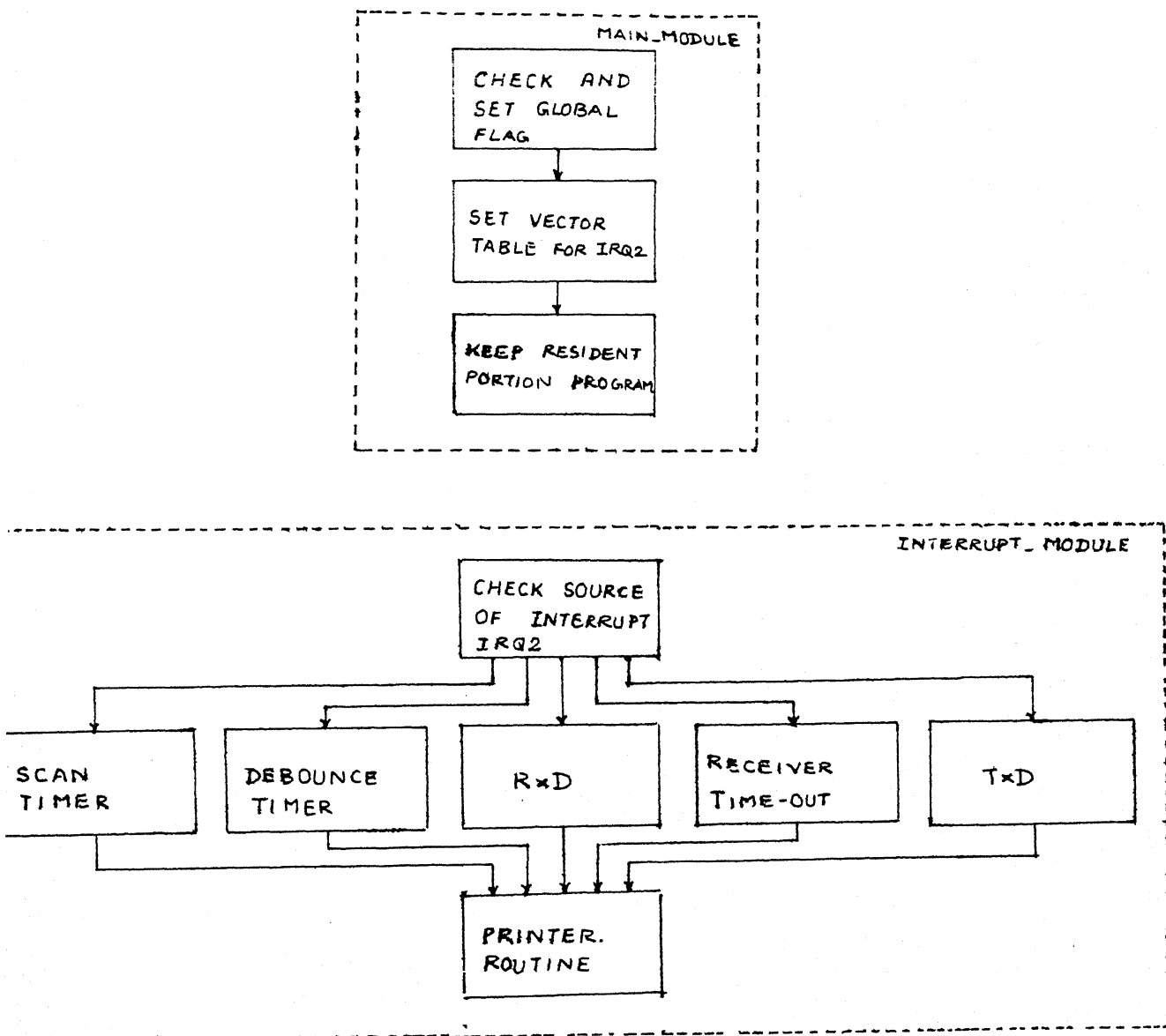


FIG. 3.10: BLOCK DIAGRAM OF CENTRAL NODE SOFTWARE

global flag, initializes the 8256 MUART and 8255 PPI chips and various other flags, variables and arrays, initializes the vector table for IRQ-2 [INT 0AH] and finally loads the resident portion of the program into memory. The control is passed back to the COMMAND.COM and rest of the AUTOEXEC.BAT batch file is executed.

The server scans the 16 serial ports for a -RTS in sequential ordering. After every minislots time interval (10mS), CPU is interrupted which checks for active -RTS line. If after a fixed number of minislots for a particular channel is over, the server begins scanning of the next channel. If -RTS becomes active during one of the minislots time interval, then Debounce Timer Interrupt is enabled. After one minislots Debounce interrupt will occur and -RTS line is again checked to see if the line is still held active. If it is held active then the request for transmission is valid and receiver of 8256 MUART is enabled and -CTS line is made active, but if -RTS line had become inactive at second checking then it infers that the request was an invalid one and so server resumes back to scan operation.

Once the receiver is enabled and the time-out timer is enabled, when the byte is received the Receiver interrupt occurs. If the byte is received in error then the error variables are updated, else the received byte is stored in appropriate buffer arrays. The time-out timer will be reloaded with the old value. During each file transfer the

central node and secondary node establishes the link at a pre-defined baud rate, character length and stop bits per byte. This baud rate is selected such that the error probability is minimum and so data bytes are normally received without any errors. The control packet of packet size equal to 8 bytes is sent by secondary node at the beginning of the transmission. The control packet contains the information about the baud rate, character length and stop bits at which the data packet will be sent and so the central node will re-program its serial interface at the user defined value and data bytes are received and stored in data array.

When the time-out interrupt occurs, the central node understands it as the end of the file transmission and so disables the receiver and sends a message packet of 8 bytes at default serial interface specification. The control packet and message packet are supposed to reach the receiver with no error while the data packet can be sent at higher speed with slight increase in error.

When the first byte is received, printer requests are enabled by clearing the latch associated with 8255 PPI. Now henceforth any request occurring due to printer connected to 8255 PPI ports will be serviced by any interrupts of 8256.

A circular buffer of 60k size is used for data storage. There are two pointers, one updated by receiver interrupt while the other updated by the printer routine when it is served. The status of the buffer is kept in a 3-valued

variable. When a pointer updated by receiver interrupt catches up with the printer pointer and the buffer status will indicate it as FULL and data reception is inhibited to prevent buffer overflow. If the pointer updated by printer routine catches up with the pointer updated by receiver interrupts then the buffer status will indicate it as EMPTY and printer request will be disabled to prevent buffer underflow.

3.3.3 Secondary Node Software:

The block diagram of the Secondary Node Software is shown in Fig.3.11. This software is also made of two modules. On power-up or during software reset the MAIN_MODULE1 will be executed inside AUTOEXEC.BAT batch file. MAIN_MODULE1 checks for global flag to prevent multiple copies of the resident portion of program in memory. If it doesn't find the flag then the flag is set and vector table for IRQ-3, IRQ-4 and INT 63H will be initialized and the resident portion of the program is kept in the memory. The control goes back to COMMAND.COM and rest of the AUTOEXEC.BAT file will be executed.

When the user at the secondary node wants to print a file, a software interrupt INT 63H will be invoked. The interrupt service routine will check if any print process is going at that node. If not, then interacts with user over a keyboard and obtains the filename and user defined serial interface for data packets. The file is read at one shot and stored in a 58Kb buffer. The 8250 is programmed in the

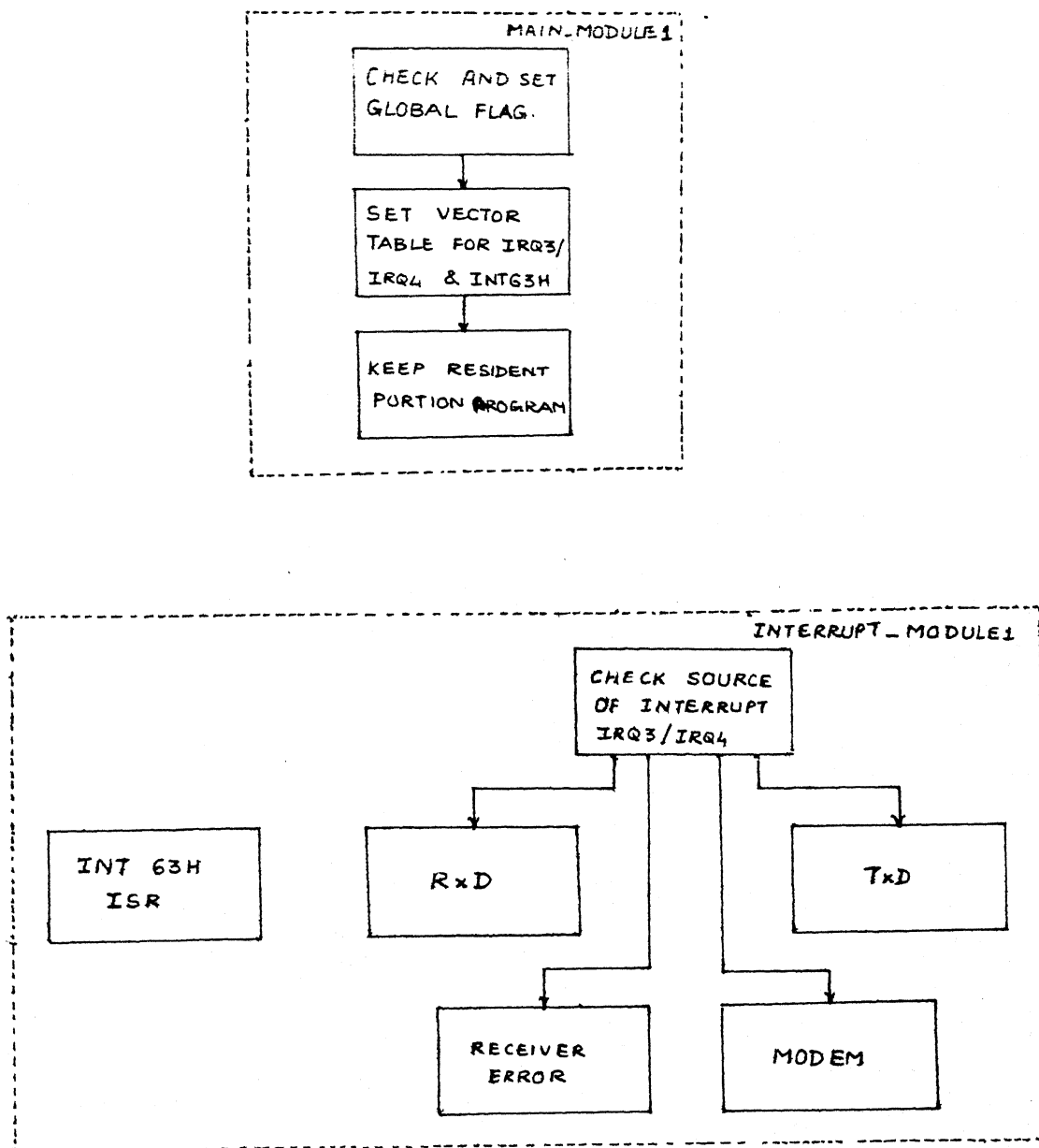


FIG. 3-11: BLOCK DIAGRAM OF SECONDARY NODE SOFTWARE

default serial specification and activate its -RTS line. When the central node acknowledges its request by sending active -CTS signal then control packet is transmitted. The 8250 will be programmed into the user specified serial interface and begins data packet transmission. When the data packet is completely received the central node will time-out after some time. The secondary node will receive message packet containing information about the number of bytes received and number of bytes were in errors. This information can be useful for network management.

The buffer is a FIFO implementation with one pointer only. The buffer size is of 58Kb, hence any file of size less than 58k will be read and transmitted without any problem. The file having size over 58Kb will have its first 58Kb read and transmitted. This restriction on file size is not so severe because generally the printable files are of size less than 58Kb.

CHAPTER 4

PRINT SERVER : A HARDWARE DESCRIPTION

4.1 INTRODUCTION:

This chapter deals with the detailed description of how the hardware for Print Server was implemented, alongwith the timing considerations. The details of specific chips like 8284-A CLOCK GENERATOR, 8256 MUART AND 8255 PPI are given in the INTEL Peripherals Handbook [8]. The logic diagram for each block is given alongwith the chip numbers as far as possible so that it becomes easy to refer the actual circuit diagram provided in the Appendix-C. The name of each of the signals is given and the Figure number where it originates is given in the bracket.

4.2 BUFFER MODULE:

4.2.1 Brief description of I/O Channel:

The I/O Channel is an extension of the 8088 microprocessor bus. It is however, de-multiplexed, re-powered, and enhanced by the addition of interrupts and direct memory access (DMA) functions.

The I/O Channel contains an 8-bit, bi-directional data bus, 20 address lines, 6 levels of interrupt, control lines for memory and I/O read or write, clock and timing lines, 3 channels of DMA control lines, memory refresh timing control lines, a channel-check line, power and ground for the

adapter. Four voltage levels : +5V DC , -5V DC , +12V DC and -12V DC are provided for the I/O cards.

The I/O devices are addressed using I/O mapped address space. The channel is designed so that 768 I/O device addresses are available to the I/O channel cards. The I/O channel is re-powered to provide sufficient drive to all 8 expansion slots, assuming two LS loads per slot.

A short description of signals that are utilized in the card is given in Table-4.1. A diagram of I/O Channel is shown in Fig.4.1 and the signals which are used has been marked in the diagram.

4.2.2 Description of Address - Data Buffering :

There are 10 address lines (A0 to A9) and 6 control lines (CLK, ALE, AEN, -IOR, -IOW and RESET) to be buffered and all of them are unidirectional in nature (output from I/O Channel), hence two buffers (74LS244) are required for buffering them. These buffers are enabled continuously and hence the buffered signals are delayed by propagation delay (18nS maximum) of the chip. As shown in timing diagram (Fig.4.3), the falling edge of ALE occurs within 15nS of the rising edge of CLK in T1 state. This falling edge of ALE is used by the system card to de-multiplex the address-data bus and so the address is available at the address lines of I/O Channel at most by 33nS with respect to (w.r.t.) the rising edge of the CLK in T1 state. As a result the buffered address bus is available in the card at most by the end of

TABLE 4.1 : I/O CHANNEL DESCRIPTION

SIGNAL	SOURCE	DESCRIPTION
CLK	Output	System Clock: It has a period of 210nS The clock has 33% duty cycle.
RESET DRV	..	This line is used to reset or initialize system logic upon power-up or during a low line voltage outage. This signal is synchronized to clock edge
A0 - A9	..	Address bits 0 to 9 : These lines are used to address I/O devices within the card. They are active high.
D0 -D7	I/O	Data bits 0 to 7: These lines provide data bus bits 0 to 7 for the processor memory and I/O devices.
ALE	Output	Address Latch Enable: This line is provided by the 8288 Bus Controller and is used on system board to latch valid address from processor on falling edge of clk
I/O CH RDY	Input	I/O Channel Ready: This line, normally high (ready), is pulled low (not ready) by I/O device to lengthen I/O cycles. Any slow device will drive it low immediately upon detecting a valid address and a read or write command.
IRQ2	Input	This line is used to signal the processor that an I/O device requires attention. An interrupt request is generated by raising IRQ2 line (low to high) and holding it till processor acknowledges.
-IOR	Output	It instructs an I/O device to drive data onto the data bus.
-IOW	..	It instructs an I/O device to read data from the data bus.
AEN	..	When it is high, DMA controller has control over the I/O channel. Hence it is used to de-gate the processor from I/O channel during DMA operation.

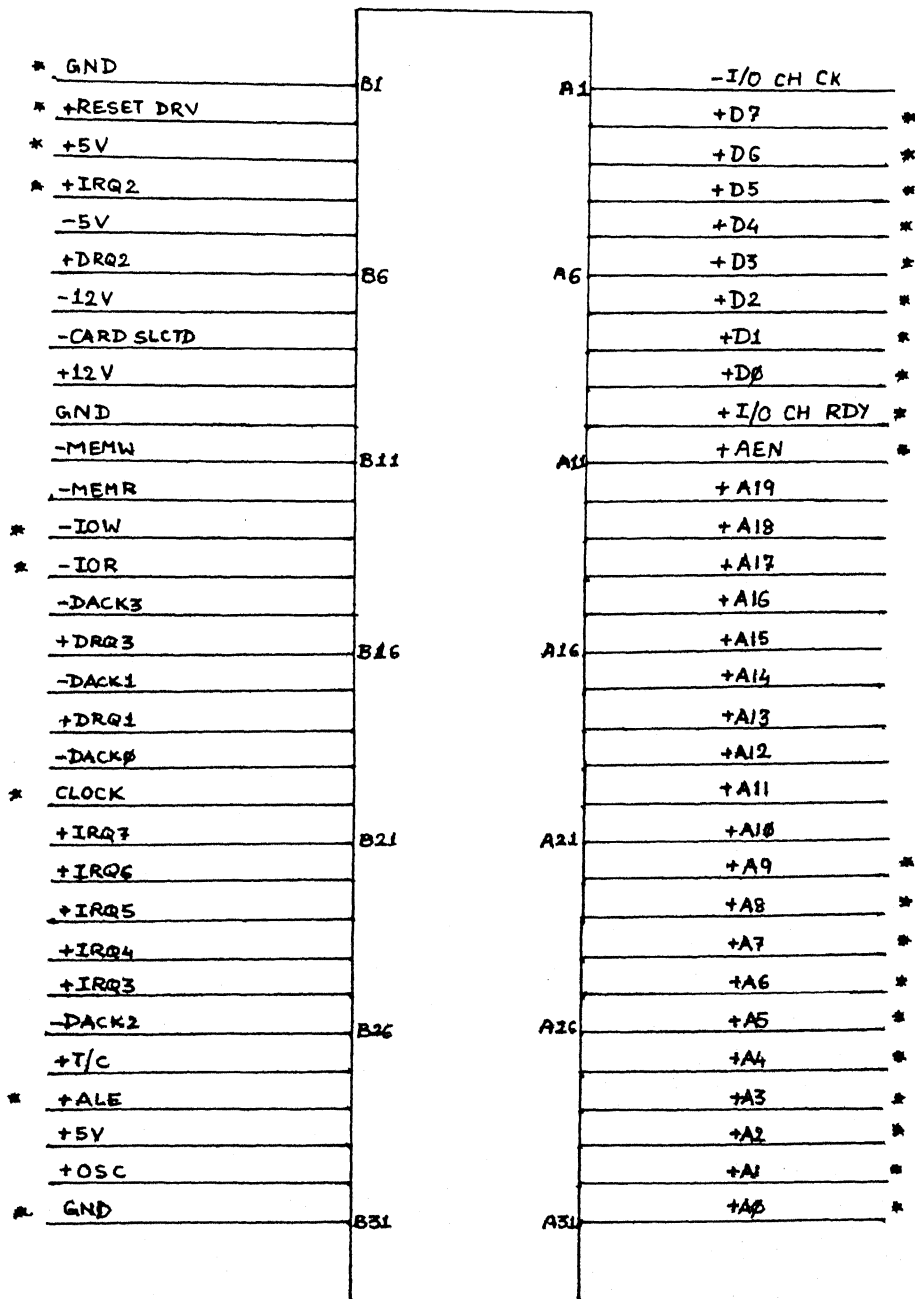


FIG-4-1: I/O CHANNEL DIAGRAM

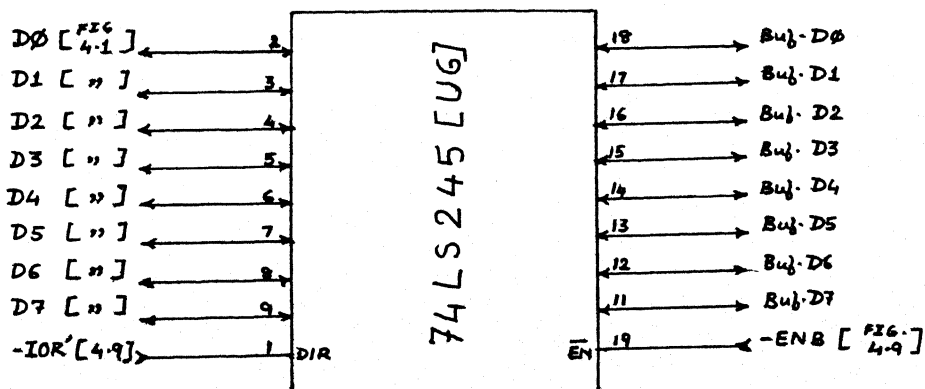
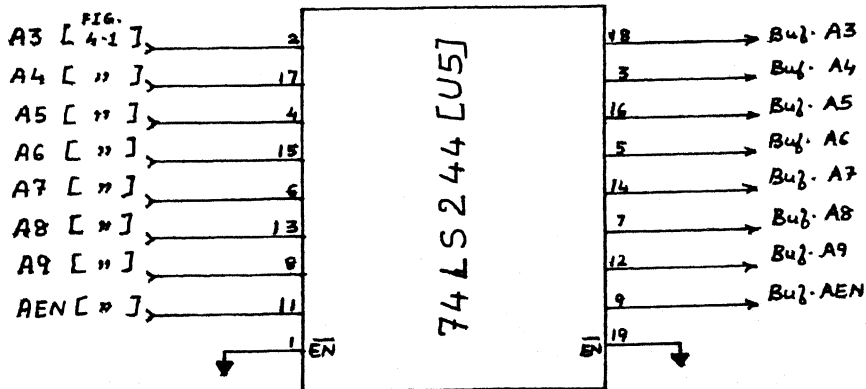
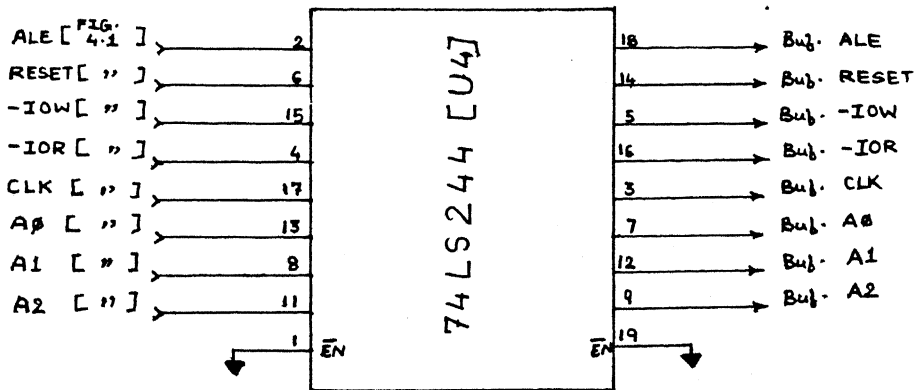


FIG.- 4.2: LOGIC DIAGRAM OF BUFFER MODULE

The buffering of data bus is not so straight-forward because the signals are bi-directional in nature and so the card can also drive this signal which can sometimes cause data contention on bus. There are 8 data lines to be buffered, hence one bi-directional buffer (74LS245) as shown in Fig.4.2 is sufficient. The direction of this buffer is controlled by a modified version of -IOR signal, namely -IOR' (discussed in Section-4.4). This signal is normally high, so the buffer is placed in the write mode for 8088 processor and thus bus contention is avoided. When a specific I/O read has to take place, this signal goes low, reversing the buffer direction such that the card can put out data onto the data bus. This buffer is to be enabled only when specific I/O read or write occurs, so the enable pin of the buffer is controlled by signal -ENB (discussed in Section-4.4) generated by -IOR, -IOW and Chip Select (-CS). Thus the card is de-gated unless I/O operation for that card is required. The relevant logic diagram is shown in Fig.4.2.

4.3 ADDRESS DECODING LOGIC:

The prototype card I/O space is restricted between 0300H to 031FH, thus 32 I/O locations are available. Normally full I/O decoding would be done, but the I/O Channel of IBM PC/XT is designed such that only 768 I/O devices are supported, hence a partial I/O decoding is done. The I/O card in the I/O channel should not respond when DMA Controller is controlling the system bus, so we use AEN

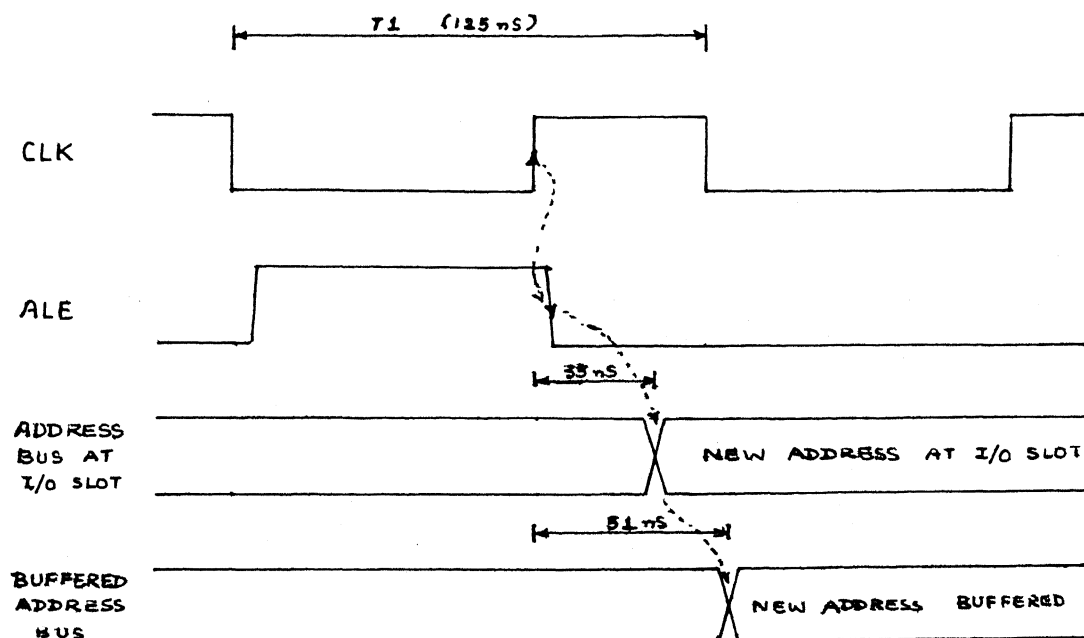


FIG.- 4.3 : TIMING DIAGRAM-1

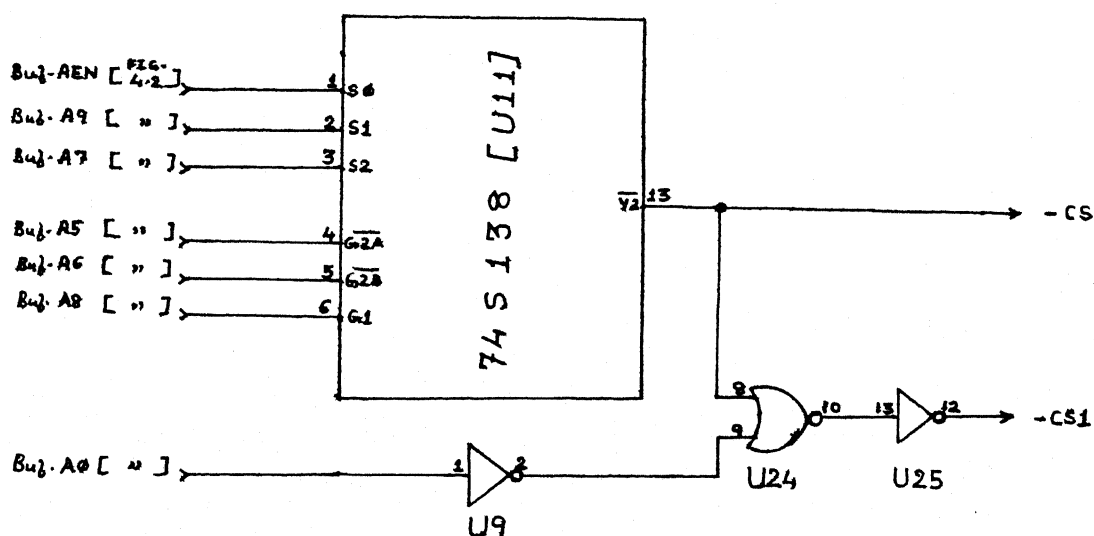


FIG.- 4.4 : LOGIC DIAGRAM OF ADDRESS DECODING MODULE

signal to de-gate the card during DMA operations. The address decoding scheme is shown below:

	AEN	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
300H - 31FH	0	1	1	0	0	0	x	x	x	x	x

There are 6 signals to determine the chip selection. A 3 to 8 line decoder (74S138), shown in Fig.4.4, chip has 3 control pins and 3 select pins to determine the decoded output, the select pins $S_2 = A_7 = 0$, $S_1 = A_9 = 1$ and $S_0 = AEN = 0$ are used so the decoded output will be available on -Y2 pin. The decoding is enabled when -G2A, -G2B and G1 are active, hence -G2A = $A_5 = 0$, -G2B = $A_6 = 0$ and $G1 = A_8 = 1$ are used. This chip select -CS is used for 8256 MUART, while 8255 PPI has odd I/O location from 301H to 307H so the chip select of 8255 PPI (-CS1) is generated by -CS and -A0 signals. The logic diagram is shown in Fig.4.4.

As shown in timing diagram (Fig.4.6) the -CS signal is obtained at most by 62nS (51nS for buffered address + 11nS for propagation through decoder chip) w.r.t. to the rising edge of CLK in T1 state, thus it becomes available somewhere in T2 state.

4.4 MULTIPLEXED ADDRESS/DATA MODULE:

This module consists of three sub-modules, which are separately discussed in this section. The sub-modules are for the following:

- a) Wait State Logic
- b) Generation of Control signals for 8256
- c) Address-Data Multiplexing

4.4.1 Wait State Logic:

It is required to insert two wait states whenever the -CS occurs. The wait state generator is obtained by using one D Flip-Flop (74S74) and few other gates. The D Flip-Flop (f.f.) of Schottky version is used to reduce the propagation delay. The f.f.-1 of Chip-U1 as shown in Fig.4.5 is cross-coupled with f.f.-2 of the same chip. The f.f.-1's CLR input and f.f.-2's PRESET inputs are connected such that outputs of f.f.-1 (i.e. -Q1) and f.f.-2 (i.e. -Q2) are set and reset respectively at each machine cycle when ALE or RESET occurs. Normally -CS is inactive, so D-input of f.f.-1 (i.e. $D1 = \text{CS} * Q2$) is low and as a result I/O CH RDY line (i.e. -Q1) stay high and no extra wait states are inserted. When -CS occurs, then I/O CH RDY line goes low for roughly two clock states as is explained in step-wise form in Table-4.2 with the help of timing diagram shown in Fig.4.6.

If -CS goes low in $T2$, then it inserts two waits by lowering I/O CH RDY line during middle of $T3$ and holding it in $TW1$. If -CS is high in $T4$, no wait states are inserted.

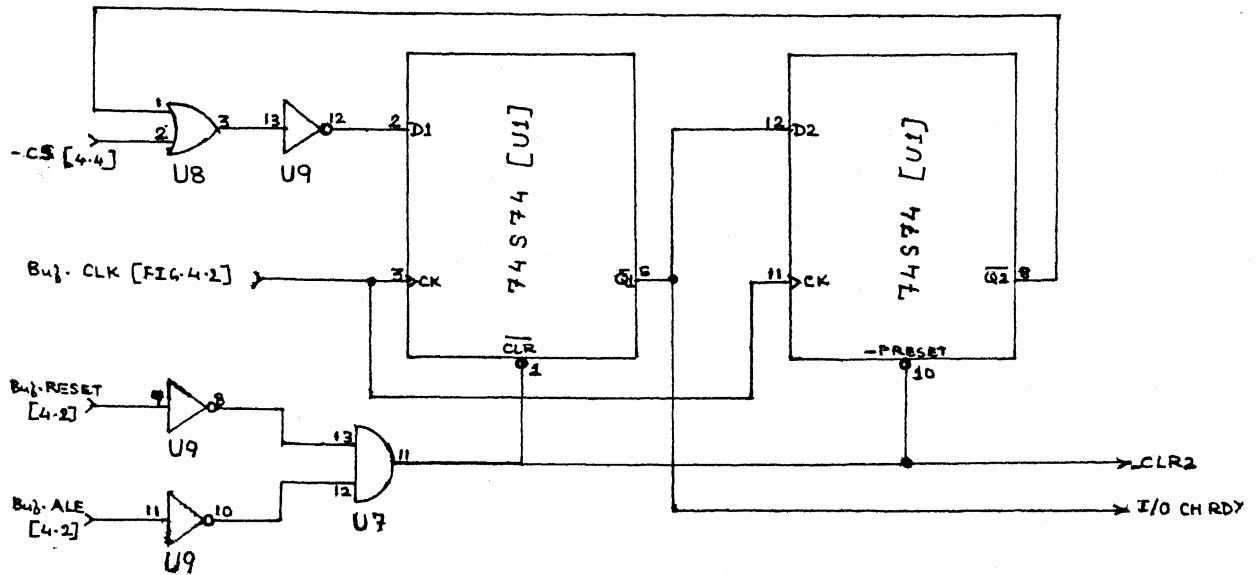


FIG.- 4.5: LOGIC DIAGRAM OF WAIT STATE CIRCUIT.

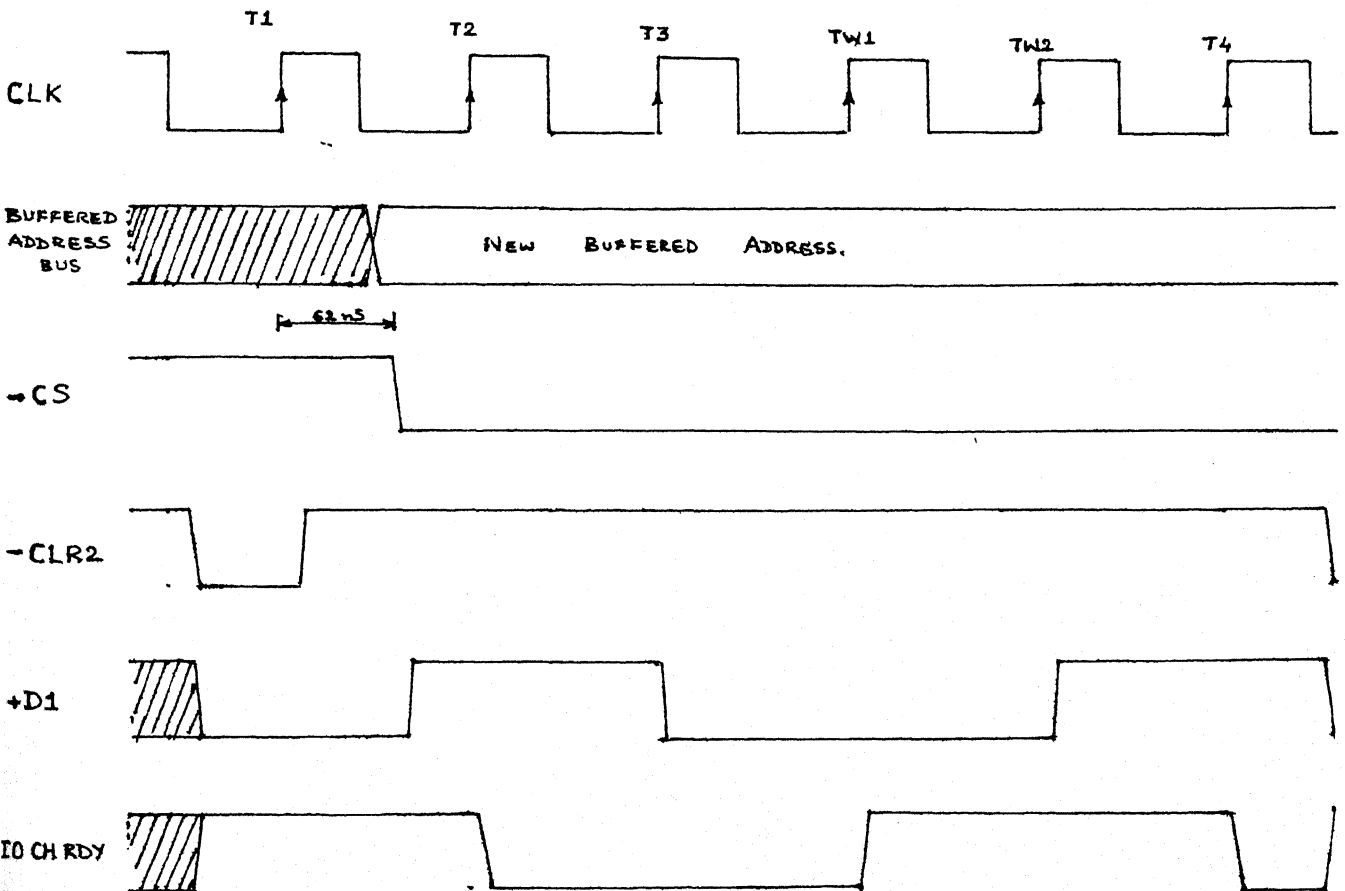


FIG.- 4.6: TIMING DIAGRAM -2

4.4.2 Generation of Control Signals of 8256 MUART:

The 8256 MUART's bus interface uses the standard bus control signals which are compatible with all INTEL peripherals and microprocessors. The $\overline{\text{CS}}$ typically derived from an address decoder is latched along with the address on the falling edge of ALE. As a result, chip select does not have to remain low for entire bus cycle. However the data bus buffers will remain tri-stated unless a read or write signal becomes active while chip select has been latched in low.

In this card the falling edge of ALE, available from the I/O Channel is lost by the time the buffered address becomes stable. Chip Select ($\overline{\text{CS}}$) becomes low at most after 63nS while the ALE goes low within 15nS w.r.t. to the rising edge of CLK in T1 state. Hence ALE has to be regenerated to meet the 8256 MUART specification. A D-f.f. (74S74 - Chip U2) is used to generate ALE. The buffered ALE signal is passed through a delay circuit consisting of few AND and NOT gates so that the signal set-up time is maintained. The D-f.f. latches the delayed ALE signal by the system CLK. The D input of the f.f. is high at the rising edge of the CLK in T1 state, so the output of f.f. (ALE') goes high. During the rising edge of CLK in T2 state the ALE is already low, therefore the D input is low and the output of f.f. goes low. Thus the regenerated signal ALE' has a falling edge occurring at 135nS w.r.t to the rising edge of CLK in T1 state. As shown in timing diagrams and discussions

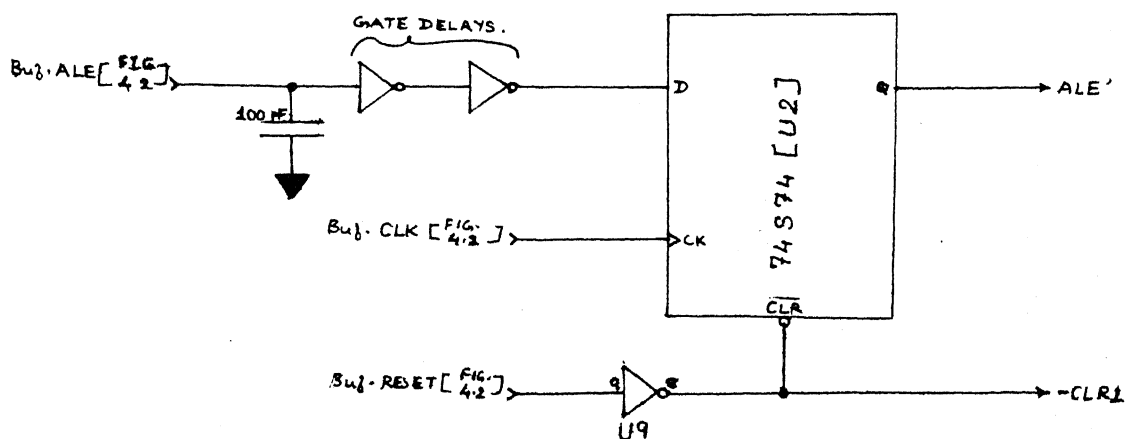


FIG. 4.7: ALE GENERATION CIRCUIT.

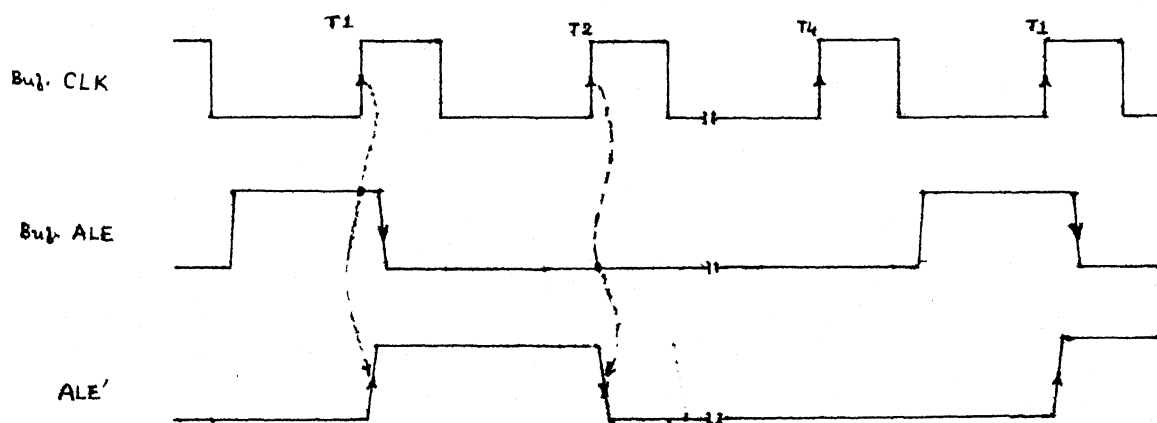


FIG. 4.8 TIMING DIAGRAM-3

from previous sub-sections, it becomes clear that -CS becomes active before falling edge of ALE' occurs and hence ALE' can latch -CS properly meeting the 8256 MUART specification. The circuit and timing diagram for ALE generation is given in Fig.4.7 and Fig.4.8 respectively.

The data buffer is to be enabled only when there is a I/O read or write request on the card and whenever the -CS is active. The -IOR or -IOW becomes available by the middle of T2 state but the falling edge of ALE' is yet to occur. Hence if data buffer is enabled at that moment, then 8256 MUART will latch data instead of address into its address latch, so we have to delay the buffer enabling till the beginning of T3 state. If -IOR or -IOW is applied directly to 8256 MUART then ALE to RD/WR set-up will not be met, hence -IOR and -IOW are delayed such that they become available to 8256 MUART only in T3 state. The minimum pulse width of RD/WR of 8256 MUART is 200nS . If extra wait state is not inserted then the above RD/WR timing could not be met, hence I/O CH RDY is pulled low in T2 state.

A Quad D-f.f. (74S175) is having 4 D inputs and 1 clear input is used to generate the delayed buffer enable signal, -RD and -WR to 8256 MUART as shown in Fig.4.10. The D-f.f. is clocked on the falling edge of the CLK. The outputs of f.f. latches the latest information on the D-input lines, as shown in timing diagram (Fig.4.10). The output signal ENB is low during T1 and T2 state and is applied to Chip-U10 (74LS244) as shown in Fig.4.11 while -ENB is low during T3 ,

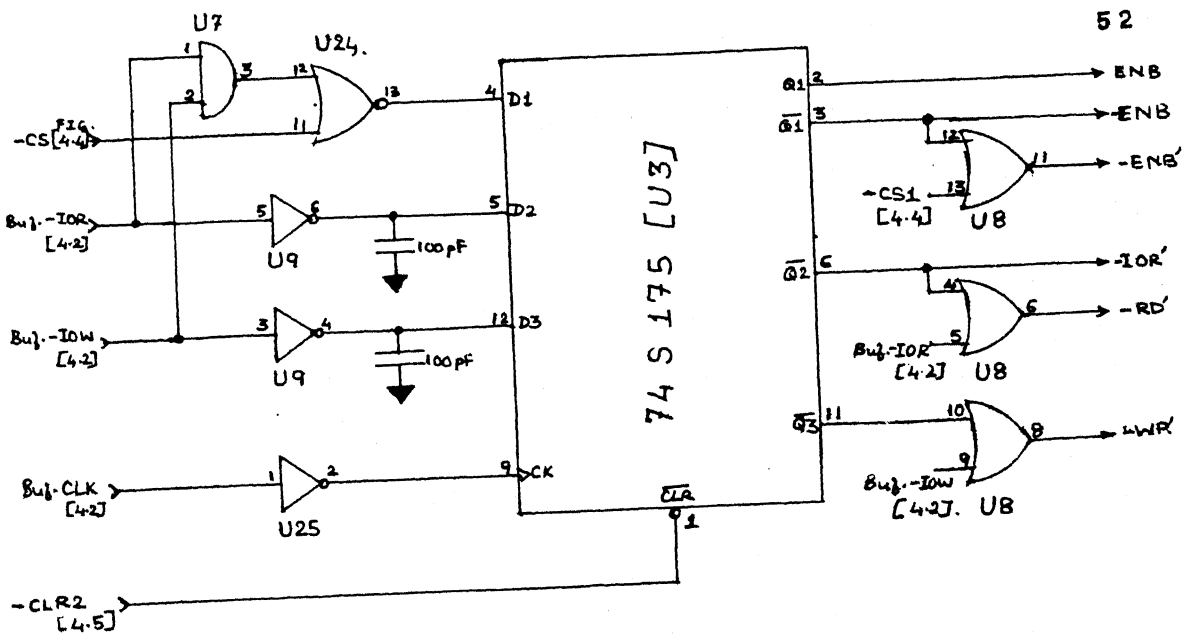


FIG.- 4.9: GENERATION OF READ, WRITE & BUFFER ENABLE SIGNALS.

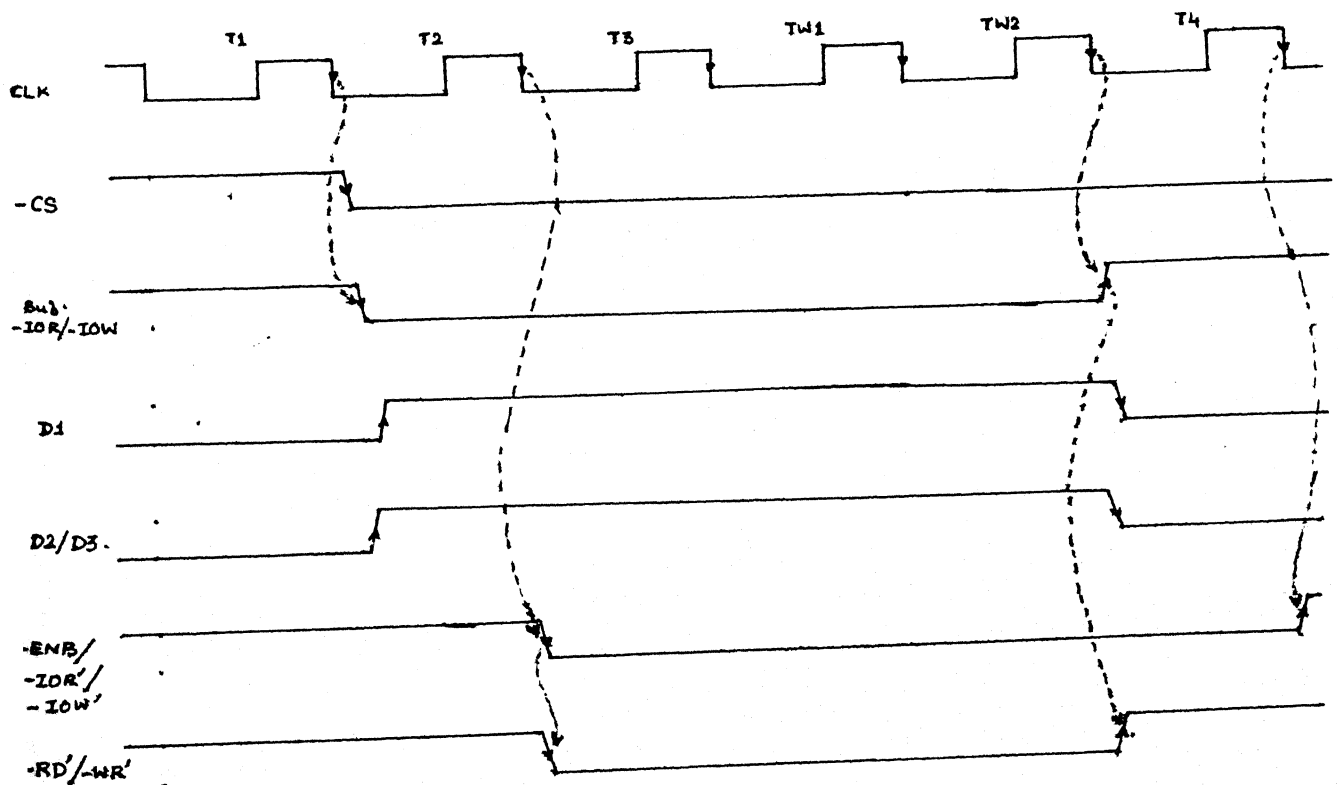


FIG- 4.10: TIMING DIAGRAM - 4.

TW1, TW2 and T4 is applied to data buffer Chip-U6 (74LS245) and $\text{-ENB}'$ (is logical ORing of CS1 and -ENB) is applied as output enable to data buffer Chip-U12 in Fig.4.11. The explanation for the above is given in sub-section 4.4.3.

The data is read by the processor or written into the external device by processor on the rising edge of read or write signal. As a result the rising edge of read or write is required. Hence the read signal applied to 8256 MUART is generated using logical ORing of -IOR and delayed version of it generated by Quad D-f.f.(i.e. -Q2). Similarly write signal is generated by ORing -IOW and its delay version (i.e. -Q3). The -RD or -WR signal applied to 8256 MUART has its falling edge corresponding to the falling edge of the delayed version of the signal from D-f.f. and rising edge corresponding to rising edge of -IOR or -IOW , thus taking care of ALE to RD/WR setup time and also maintaining the valid data edge for data transfer.

4.4.3 Address-Data Multiplexer:

As shown in Fig.4.11 the unidirectional buffer Chip-U10 is switched by signal ENB, while the bi-directional data buffer Chip-U12 is enabled by $\text{-ENB}'$ and Chip-U6 by -ENB . During an I/O request for 8256 MUART, ENB will be low in T1 and T2 state and buffered address A0 to A4 will be available to 8256 MUART address-data lines after two buffer delay. In T3, TW1, TW2 and T4, ENB is high and -ENB is low which results into Chip-U10 being OFF and Chip-U6 and U12 being ON and data becomes available to 8256 MUART onto its

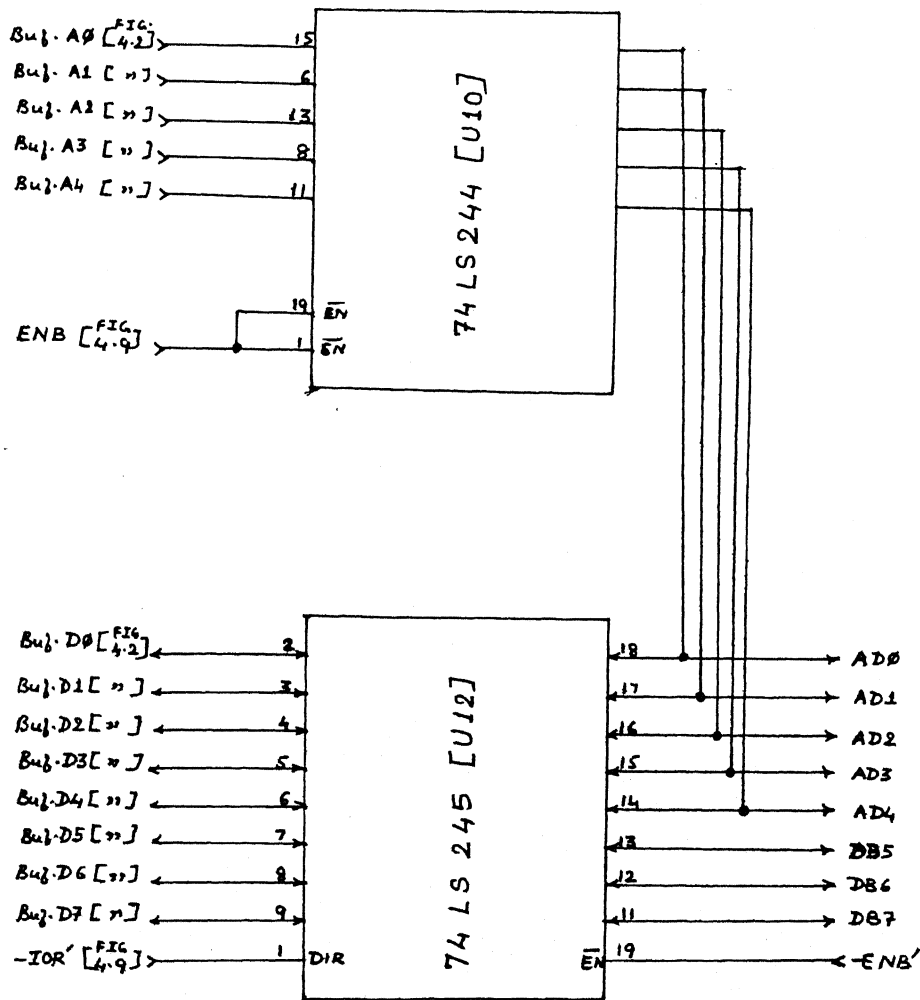


FIG.- 4.11 ADDRESS- DATA MULTIPLEXER.

address-data lines. Thus address and data have been multiplexed onto the same lines.

If there is an I/O request for 8255 PPI, -CS1 goes low in T2 state resulting into -ENB being low and -ENB1 being high from T3 to T4 state, thus only Chip-U6 is enabled but Chip-U12 will stay tri-stated and data contention between 8255 and 8256 is avoided.

4.5 MAIN MODULE OF THE CONTROLLER:

This module can be divided into four basic constituents, as follows:

- a) Clock generator circuit for serial I/O
- b) MUART and its associated signals
- c) PPI and its associated signals
- d) Interrupt generation logic

4.5.1 Clock Generator Circuit for Serial I/O:

The clock generator chip 8284-A requires a frequency source, so a crystal oscillator of 6.144 MHz is connected across X1 and X2 pin of 8284-A. The clock output is available on CLK pin (one-third of input frequency) and on PCLK pin (one-sixth of input frequency). The PCLK is connected to 8256 MUART's CLK pin to feed the five timers and baud rate generator. This clock is asynchronous to the microprocessor's clock. The clock pre-scaler inside 8256 MUART is a programmable divider which normalizes the internal clocking frequency for the timers and baud rate generator to 1.024 MHz. Here since the PCLK is at 1.024 MHz so the pre-scaling factor is set at 1.

4.5.2 MUART and its associated signals:

The four commonly used peripheral functions contained in MUART are: a) Two 8-bit parallel I/O ports

b) Five 8-bits counters/timers

c) 8-level priority interrupt controller

d) Full duplex, double-buffered serial UART

The detailed chip description is given in Ref-[7]. Muart provides all the four commonly used peripheral functions in a 40-pin DIP while retaining direct register addressing, so uses a multiplexed address/data bus. The de-multiplexed address/data bus available on I/O channel is again multiplexed as described in section-4.4.3. The chip select signal -CS is derived from address decoder (section-4.3) is latched alongwith the address on the falling edge of re-generated ALE' (section-4.4.1). The data bus will remain tri-stated unless a -RD or -WR is activated (section-4.4.2) while -CS has been latched in low and address line A0 is also low. When a RESET line gets activated the MUART is placed in a known initial state.

Muart provides 5 lines for serial I/O. The two basic serial I/O lines RxD and TxD are used in normal fashion. There are two lines RxC and TxC for external clocking of serial data, this mode is not used. The -CTS line of MUART is held low and so MUART is always active for transmission.

MUART provides two parallel I/O ports, Port-1 is bit programmable for input or output while Port-2 is nibble programmable as well as input or output byte handshake mode.

In this card Port-2 is programmed into output byte handshake mode with two pins of Port-1 acting as handshake lines. Out of the remaining 6 lines of Port-1, one line is used as input working as -RTS line from secondary nodes, while one line is used as output working as -CTS line for secondary nodes and remaining four lines are used for serial channel selection.

MUART has five 8-bit programmable counters/timers, with Timers-2 and 4 and Timer-3 and 5 can be cascaded into 16-bit timers/counters. In this card Timer-1 is used as a 8-bit timer for debouncing the -RTS line. Timer-2 and 4 are cascaded to form a 16-bit timer to do scanning of -RTS line. Timer-3 and 5 is again cascaded as a timer for time-out operation if the byte is not received. The interrupt function is described in section-4.5.4.

4.5.3 PPI and its associated signals:

INTEL 8255A contains a control register and three separately addressable ports, denoted by A,B and C. 8255 is accessed by the signal on -CS pin and the direction of the access is determined by -RD and -WR signals. In this card, the 8255 has its data bus connected to data lines of data buffer Chip-U6. The four addressable registers are located at odd locations starting from 301H to 307H, hence -CS1 signal is connected to -CS pin of 8255 and register will be referred using address lines A1 and A2.

Port-A and B are connected as output port in Mode-0. They act as a parallel port for CENTRONICS Interface Line

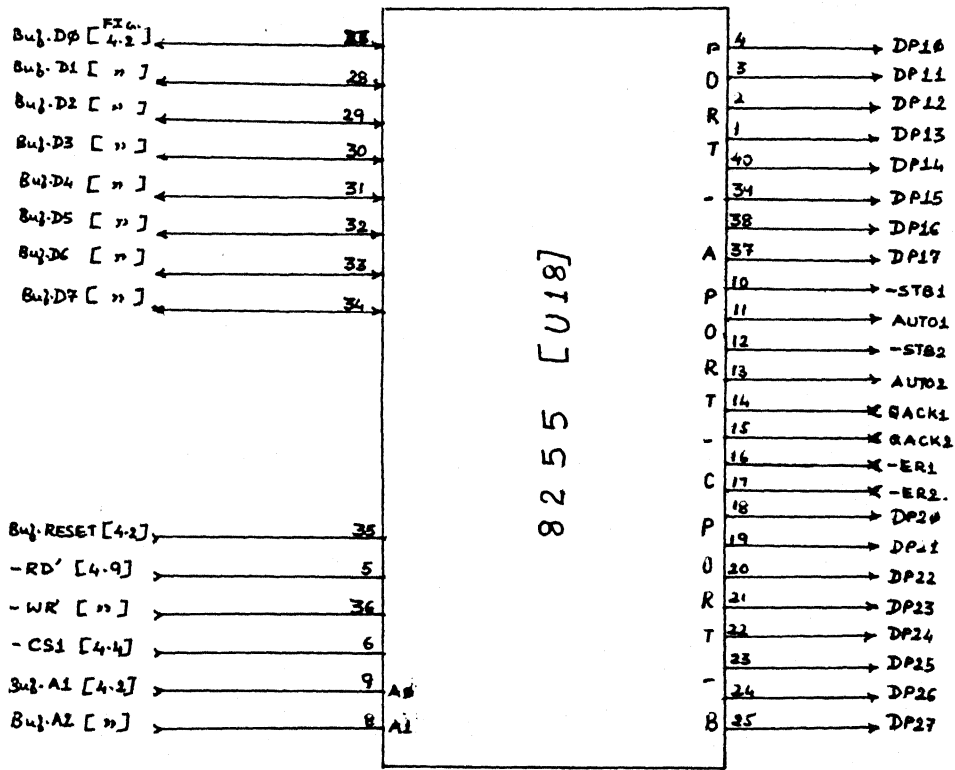


FIG. 4.13 : 8255 PPI CHIP INTERCONNECTION.

Printers. Port-C is used as control and status port for above printers. The upper nibble of Port-C is used as control or output port with two lines used for sending the STROBE pulse to the printers. This pulse is generated using bit set/reset function. The other two output pins are used for generating automatic Form-Feed. The lower nibble of Port-C is used as status or input port. The two lines contain the level of ACKNOWLEDGE pulse and the other two lines contain the information of printer error condition.

4.5.4 Interrupt Generation Logic:

The 8256 has INT and -INTA signals to interrupt the CPU and receive the CPU's acknowledgment to the interrupt request. On the I/O Channel there is no -INTA line and neither is CAS0-2 lines by which the 8256 can be connected in slave mode. Hence the MUART's INT line is connected to IRQ2 line of I/O Channel and -INTA is tied to Vcc. The interrupt service routine will read the Interrupt Address Register to perform interrupt acknowledge to CPU.

The CENTRONICS Interface Line Printers generates acknowledge pulse for requesting the next byte to be transferred. A D-f.f. (74LS74) is used to latch this acknowledge pulse and store the level in one of the input pins of Port-C of 8255. The D-input of the f.f. is tied to Vcc and Acknowledge pulse is connected to clock line of the f.f. so the output of f.f. is high whenever an acknowledge pulse occurs. The printer routine which serves the data transfer request will generate STROBE pulse on one of the

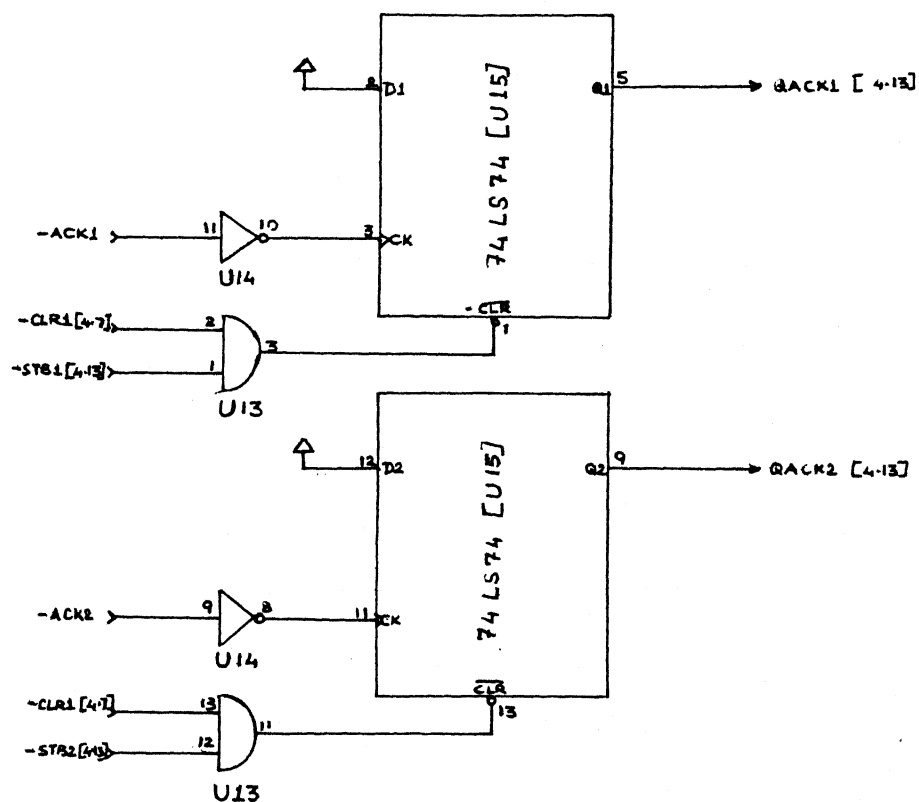


FIG.- 4.14 : CENTRONICS PRINTER REQUEST GENERATION.

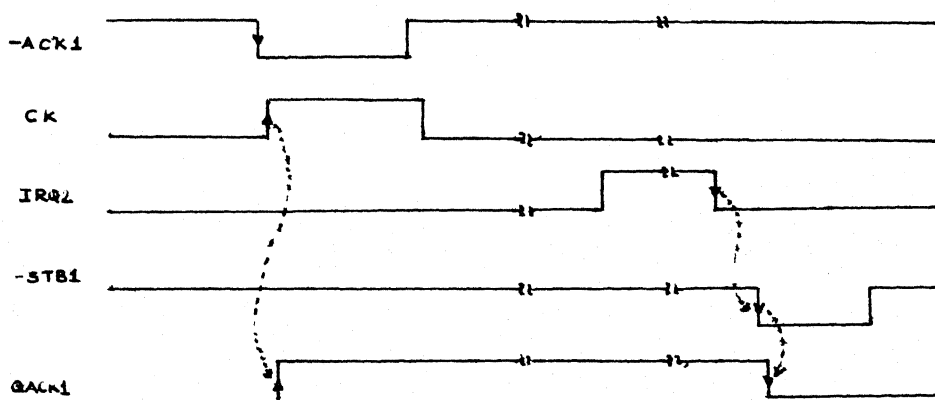


FIG-4.15 : TIMING DIAGRAM-5

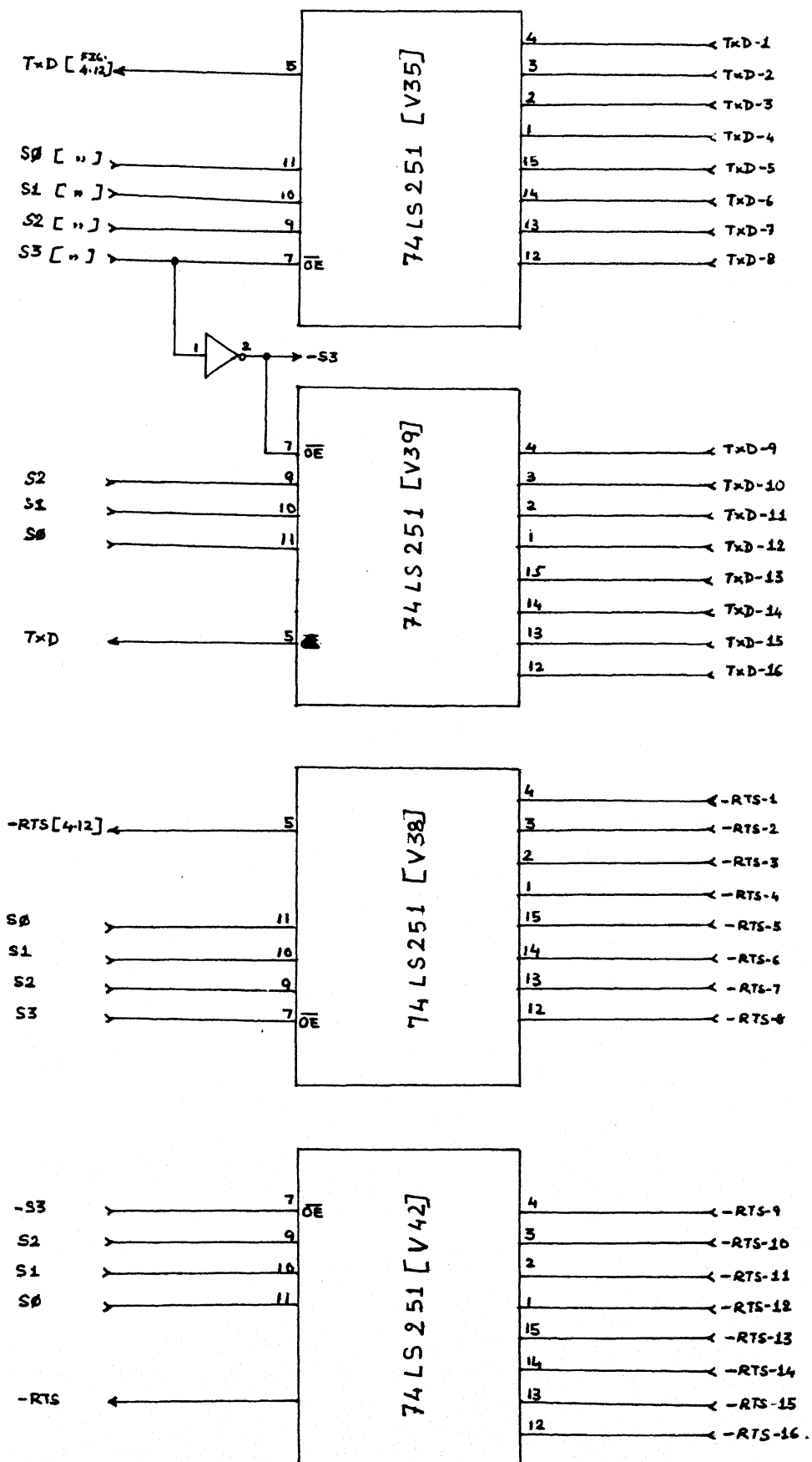


FIG.- 4-16 (A): MULTIPLEXING LOGIC FOR SERIAL I/O.

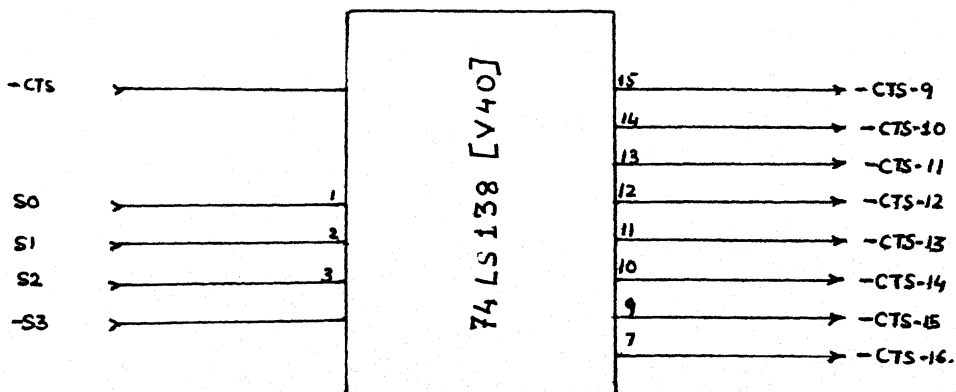
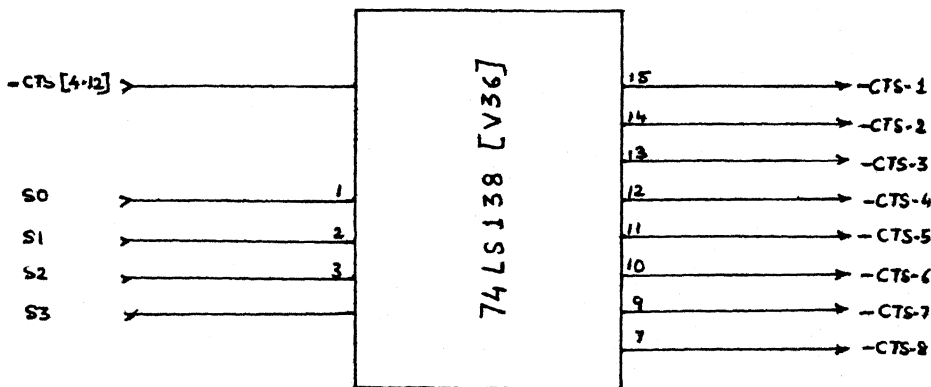
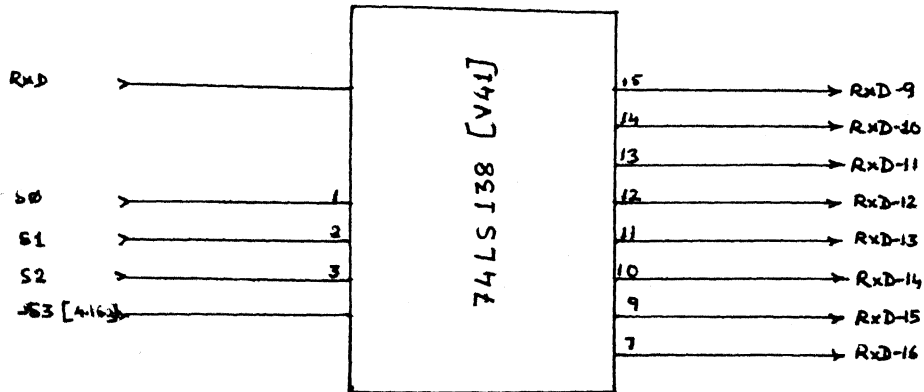
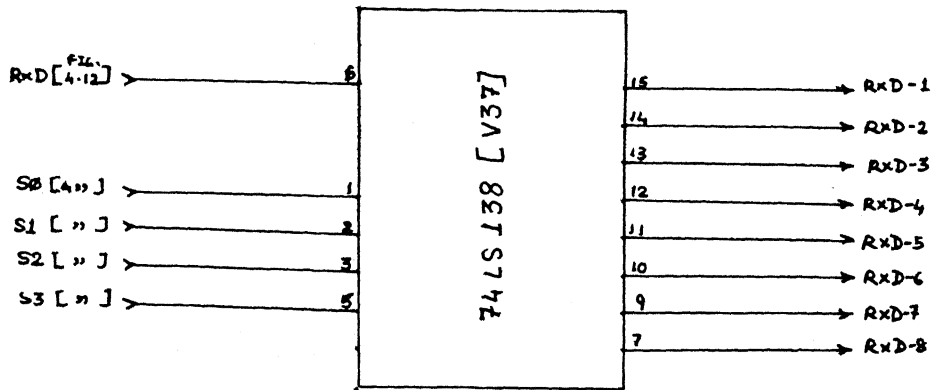


FIG.- 4.16 (B): DEMULTIPLEXING LOGIC FOR SERIAL I/O.

The 8256 MUART transmits data on its TxD pin. This data can be sent to any one of the secondary nodes on its RxD line. As a result the TxD line of 8256 MUART is de-multiplexed using two 1 to 8 DEMUX. (74LS138). The resultant signals are connected to respective RxD lines of secondary node. Similarly the central node transmits -CTS signal on one of the output port pins of Port-1 of 8256 MUART. This signal is de-multiplexed using two 1 to 8 DEMUX. and resultant signals are connected to respective -CTS lines of secondary nodes.

The multiplexing and de-multiplexing logic for the 4 serial lines are addressed using 4 output lines of Port-1 of 8256 MUART. The signals S0-S3 decides which of the 16 channel is to be selected, thus 4 select lines acts as a 16-way switch.

4.7 LINE DRIVER AND RECEIVER MODULE:

4.7.1 Parallel I/O Line Drivers:

The Line Printer using either DATAPRODUCTS or CENTRONICS Interface are based on TTL level signaling. These signals are sent on a flat ribbon cable from the port to the printer, the signals get deteriorated. To prevent the deterioration of the signals line drivers (INTEL 8282) are used. The STB line of line driver is held high so that it works as a transparent buffer. INTEL 8282 is preferred over uni-directional buffer 74LS244 due to its higher driving capacity. The output enable of 8282 is held high continuously.

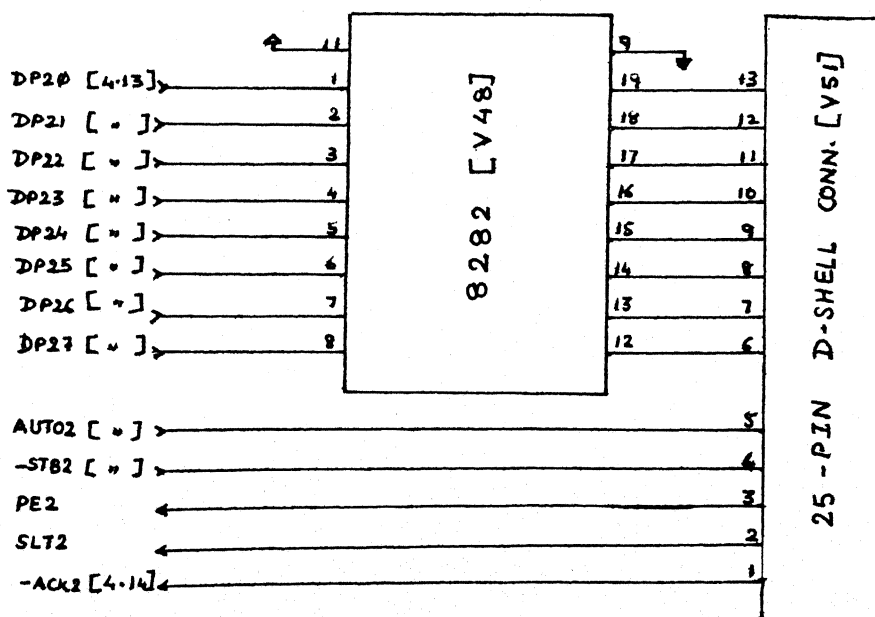
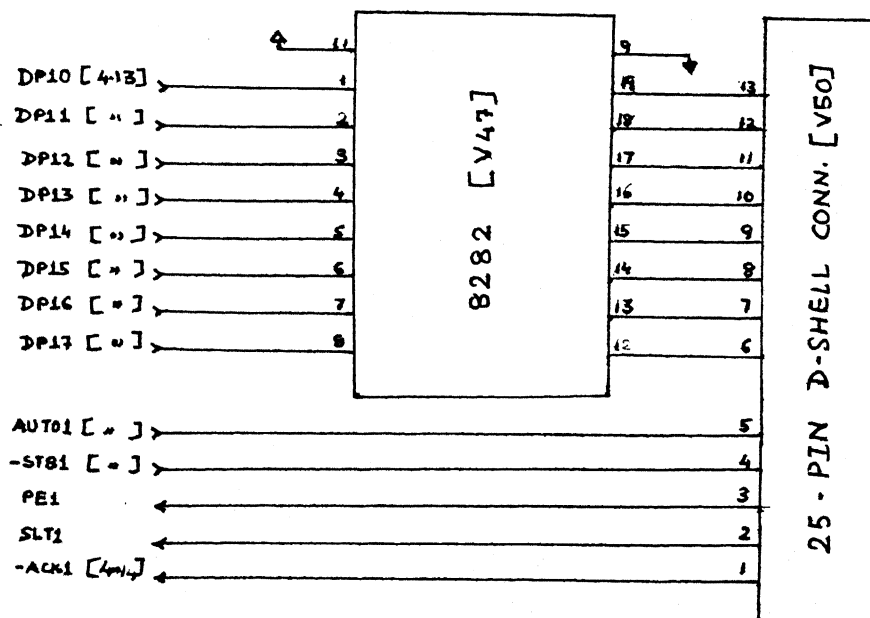
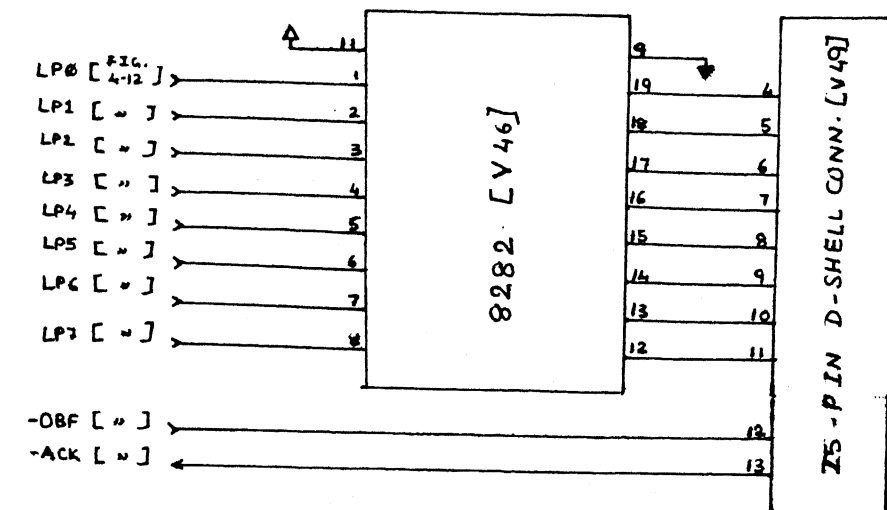


FIG. 4.17 LINE DRIVERS & CONNECTORS FOR PARALLEL I/O.

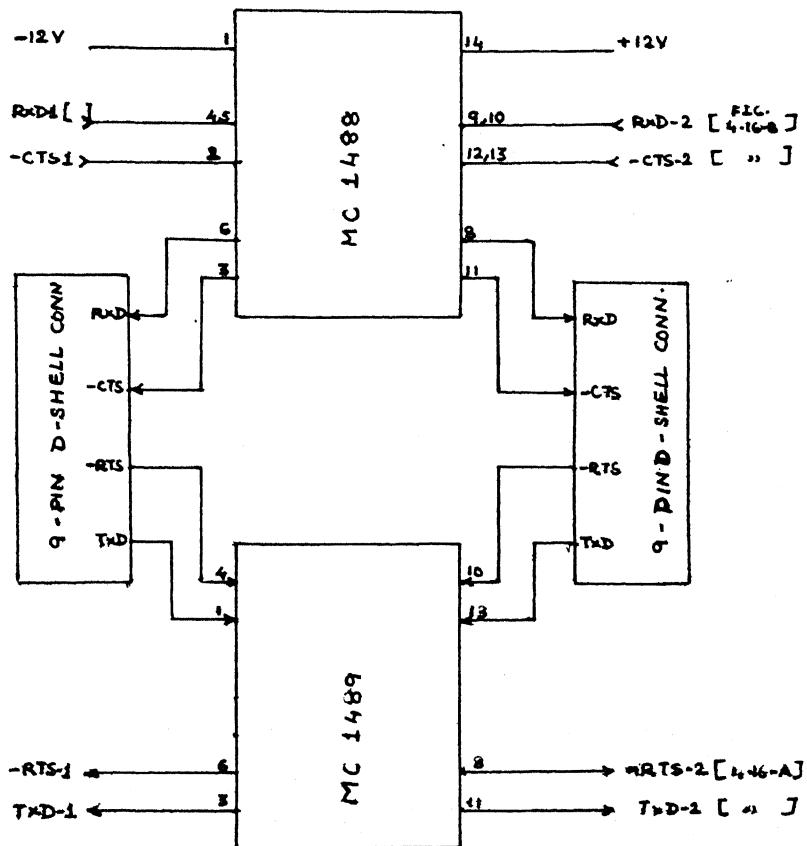


FIG.- 4.18 LINE DRIVERS & RECEIVERS FOR SERIAL I/O

4.7.2 Serial Line Drivers and Receivers:

The serial I/O link connected between the central node and secondary nodes are based on RS-232C interface. The signal level translation between TTL level and RS-232C level is required. The RxD and -CTS lines are output by central node as TTL level, these signals are translated to RS-232C level using RS-232 Line Drivers (MC 1488). The TxD and -RTS lines are available as input to central node, these signals are RS-232 level. Hence a RS-232 to TTL translation is done using RS-232 Line Receivers (MC1489). Each line driver and receiver chips support 4 level converters, hence 8 such line drivers and receivers are required.

4.8 INTER-CARD BUFFER MODULE:

In the design implementation of Print Server, two cards are required. The logic consisting of Channel Buffering, Address Decoding, Multiplexed Address-Data Module, Main Module of the controller are implemented on prototype card and is placed inside the PC using one of its I/O Channel. The logic for MUX./DEMUX. of serial interface and Line Driver and Receiver Module is implemented in the second card which is placed outside the PC. The inter-communication between the cards is obtained by using two 50-wire flat ribbon cable of a short length. The signals which are driven over this cable are buffered using buffers (74LS245). The circuit diagram of this module is shown in Fig.4.19.

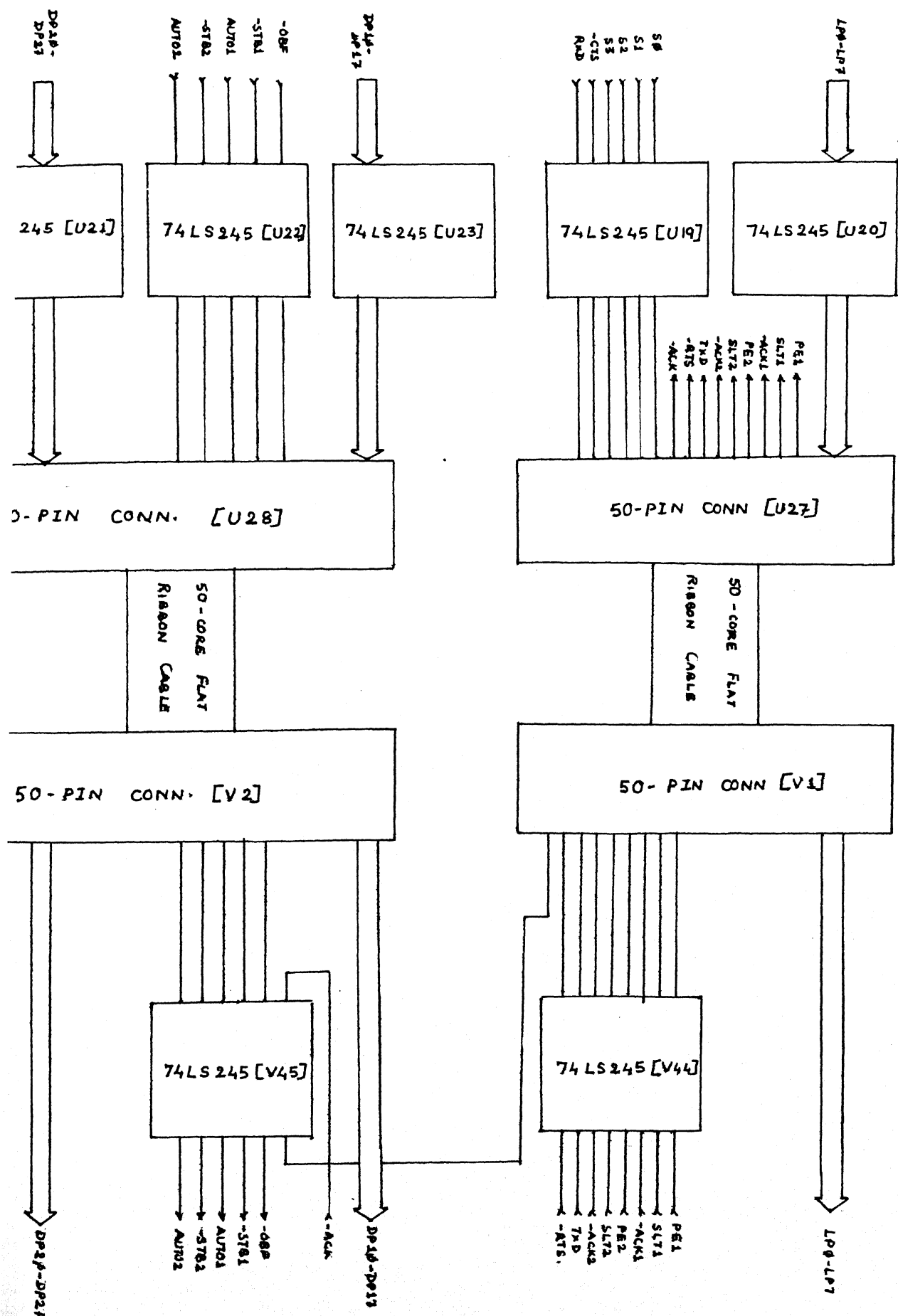


FIG.- 4.19: INTER-CARD BUFFERING CIRCUIT

CHAPTER 5

PRINT SERVER : A SOFTWARE DESCRIPTION

5.1 INTRODUCTION:

The software block diagram for secondary and central node is illustrated in Chapter-3 (Section-3.3). Here each of the procedure will be discussed in greater detail with flow-charts. In Section-5.2 the secondary node software will be discussed and in Section-5.3 the central node software is described.

5.2 SOFTWARE FOR SECONDARY NODE:

5.2.1 Introduction:

This section will discuss six main procedures forming the backbone of the secondary node software. They are as follows: a) MAIN_MODULE1

- b) Interrupt Service Routine for INT 63H
- c) Transmitter Routine
- d) Receiver Routine
- e) Modem Routine
- f) Error Routine

5.5.2 MAIN_MODULE1 Routine:

This routine has to be invoked only once for loading the program into memory and making it resident. The memory will stay occupied until the system is initialized. Hence if this program is invoked a number of times accidentally or

maliciously without some protection mechanism, then most of the memory space can go wasted and there would not be much memory space left for other programs to execute. Hence to prevent this from happening a global flag is set at a fixed memory location where it would not normally get changed till a system initialization takes place. During system initialization, the vector for INT 64H in vector table points to an IRET instruction in memory. Since it is not used normally, this interrupt can be captured and vector table location for INT 64H (0000:0190H) can be used as a fixed memory location for global flag. When the routine is invoked for the first time the global flag will have a value other than OFFFFH, hence the global flag value is set to OFFFFH. The vector table location for software interrupt INT 63H and hardware interrupts of COM ports (IRQ-3 and 4) are set pointing to appropriate Interrupt Service Routine (ISR) defined in the program. The IRQ-3 and 4 are both set to same ISR because if the system uses any of the COM port for serial transfer then also the program should work. The resident portion of the program consisting of ISRs, memory buffers for data storage, flags and variables for conditions are kept in the memory using KEEP function of DOS (Function-31H of INT 21H). The routine terminates with the message "> KEEP SUCCESSFUL."

If the global flag had been set and if this routine was invoked then it will quit directly without changing anything but with a message "> PROGRAM ALREADY INSTALLED."

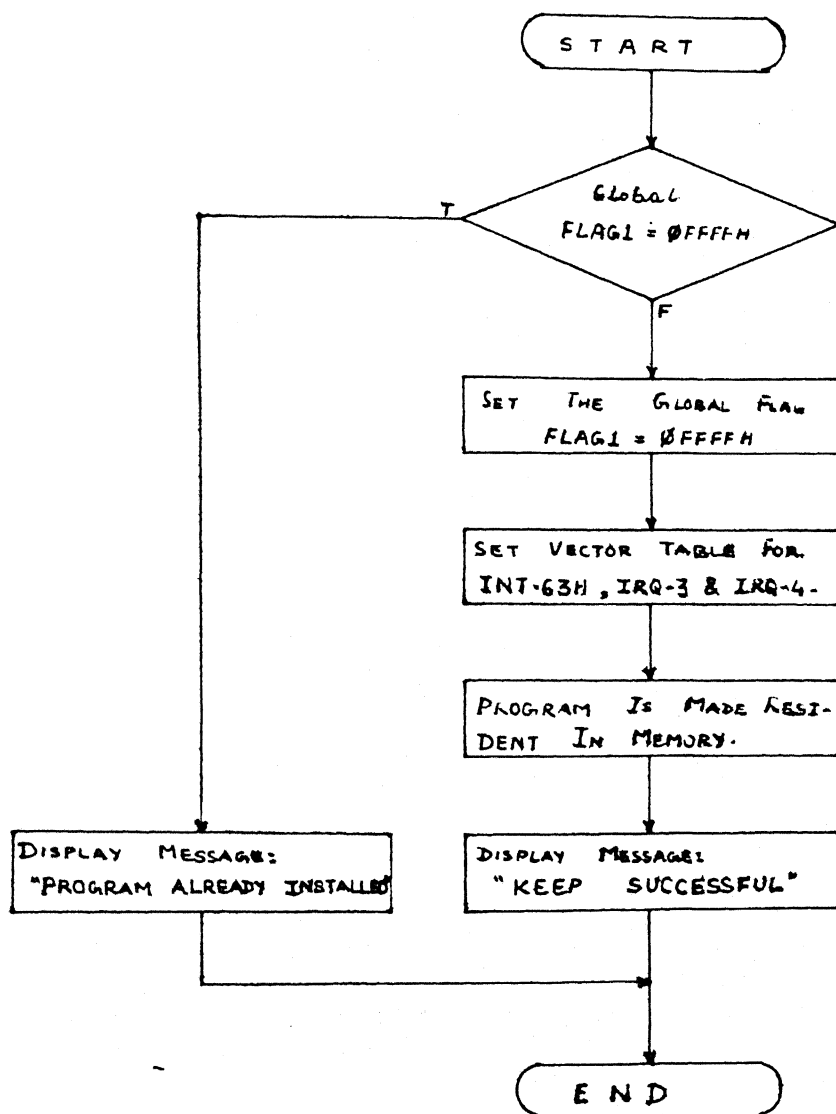


FIG.- 5.1: FLOW CHART OF MAIN-MODULE1 ROUTINE

This routine has to be invoked only once after system initialization, hence this routine will be included in AUTOEXEC.BAT batch file. After the execution of this program and the remaining commands in the batch file, the control is restored to the command processor (COMMAND.COM).

5.2.3 Interrupt Service Routine for INT 63H:

Whenever an user wants to get a print-out using the PRINT SERVER, that user will issue a command

```
> PRINTNET
```

This will invoke software interrupt INT 63H and quits. In response to the interrupt, the execution of the ISR of INT 63H will begin. A global flag FLAG2 is checked to see if another print process is still being executed or not. This global flag is stored at vector table location of INT 65H (0000:0194H). This flag is needed because the secondary node print queue is only 1 level deep, so once a file transfer begins, no other file is accepted till the previous file has been completely transferred. If the queue is empty the global flag will be set and keyboard dialogue with the user will be initiated. The user will specify the file-name and if he requires the data transmission to occur at different baud rate, character length or stop bits etc. then he has to specify accordingly. If the user doesn't specify any serial I/O specification then the data transmission will occur at the default specification applicable to the control packet transmission.

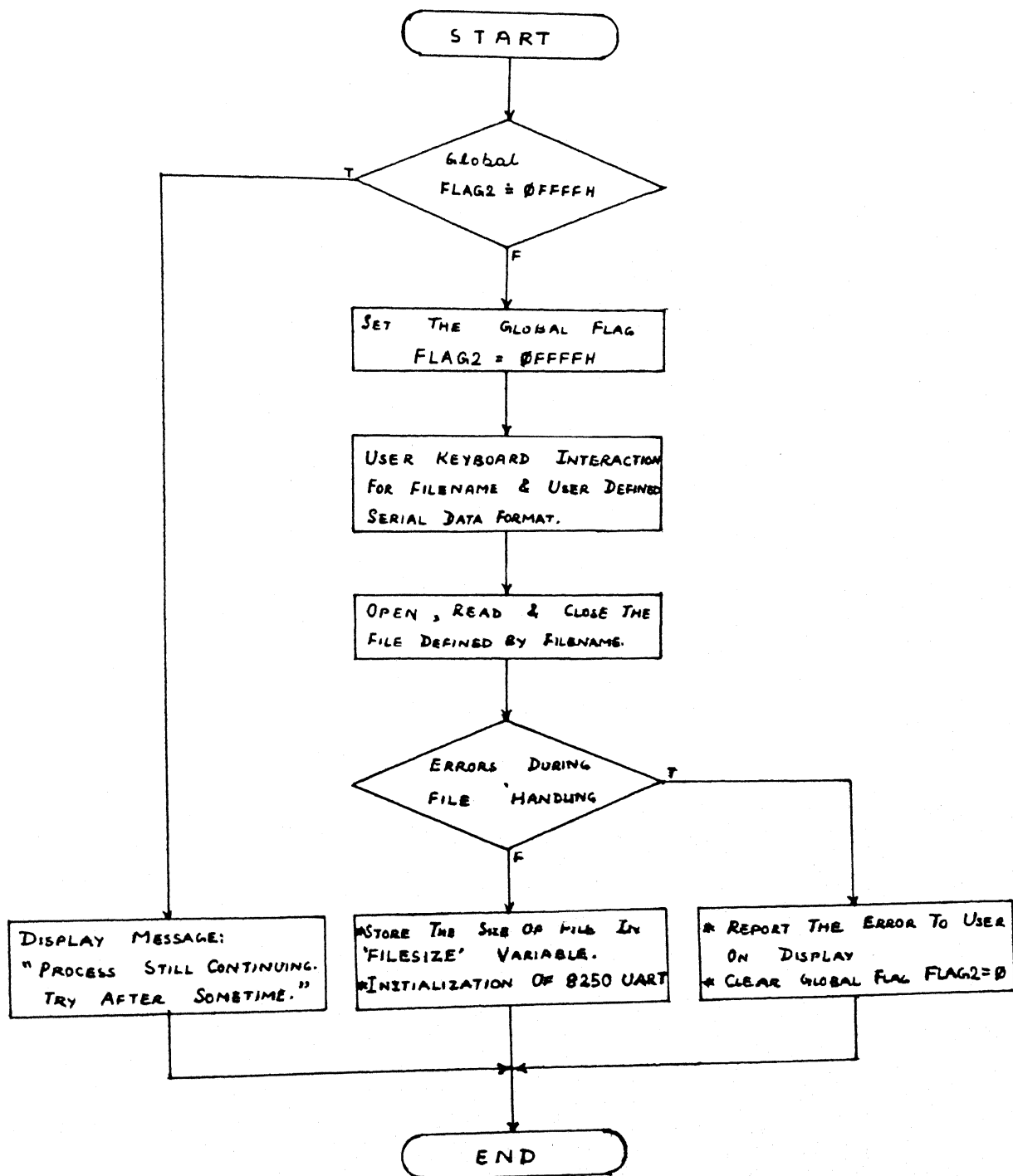


FIG. - 5.2: FLOWCHART FOR SOFTWARE INTERRUPT ROUTINE (INT 63H)

Once the file-name is specified the file is read from the disk and dumped into the data buffer specified in the memory. During the disk read any error occurring will be reported accordingly to the user and if the file can not be read then the global flag FLAG2 is cleared. As MS-DOS is not re-entrant and the print process is completely interrupt driven with a separate fore-ground task running simultaneously, it is not possible to re-load the buffer from disk during the Transmitter Interrupt Routine, so the whole file is read at one go before entering the hardware interrupt routines. Hence the file size is restricted to only the size of the memory buffer which is allocated. Here only file size of less than 58Kb will be completely printable, but if file size is greater than 58Kb then only the first 58Kb will be loaded into memory and will be printed.

If the disk read had been successful then INTEL 8250 UART is initialized and is programmed to work at default specifications. All the interrupts of 8250 UART, except Modem interrupt, are disabled. The -RTS line is made active and the interrupt service routine is completed and control is reverted back to COMMAND.COM level. The user can do his normal editing work etc.. When the file is completely transferred from the secondary node to central node then the user is informed by a flash message displayed at the bottom of the screen. The user will acknowledge the message by pressing a special key and then continuing with his work.

5.2.4 Modem Interrupt Routine:

Modem interrupt in a COM port can be generated by one of the four signals, when they change their logic levels. They are Clear To Send (CTS), Data Set Ready (DSR), Ring Indicator (RI) and Receive Line Signal Detect (RLSD). Here since 4-line serial I/O communication takes place, the only source of modem interrupt is due to -CTS level changes. The -CTS signal is used for data flow control mechanism. At the beginning, the secondary node sends -RTS line active and waits for -CTS line to become active. The central node sends -CTS line active to secondary node to begin transmission. When the buffers of the central node gets full, then to prevent the data overflow, a -CTS line will be made inactive to disable the transmission of data from secondary node for time-being. Hence a change of -CTS from inactive state to active state will indicate "begin transmission", while from active state to inactive state indicate "stop transmission". Thus a simple data flow-control is achieved.

On entering the modem interrupt routine, the state of -CTS line is checked. If the signal is in inactive state then the Transmitter interrupt of 8250 UART is disabled, but if the signal is in active state then all interrupts of 8250 UART are enabled. To make transmit interrupt operative, a dummy byte ('NUL' or 'SPACE') is loaded in transmitter buffer. When the transmitter buffer becomes empty, a transmit interrupt is invoked.

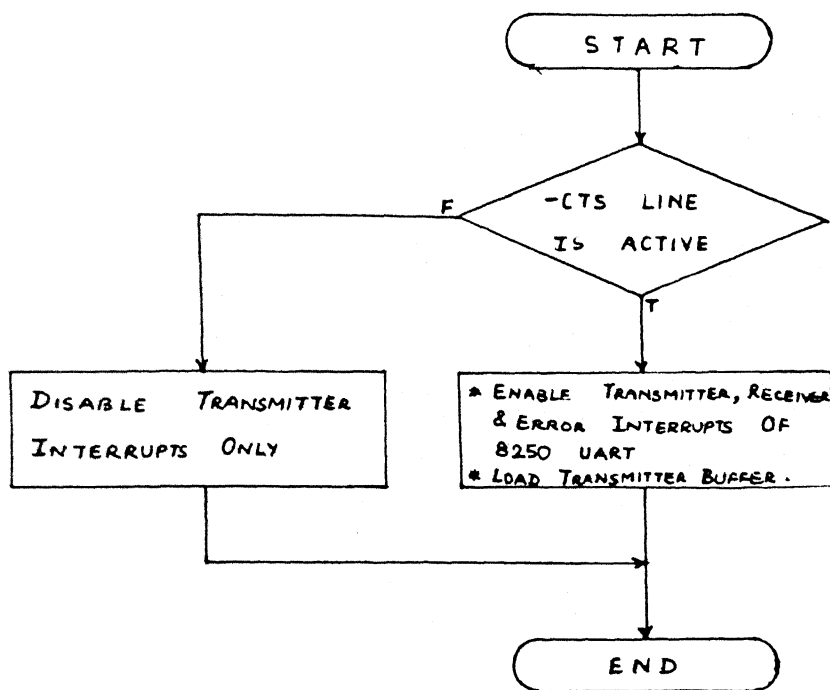


FIG.- 5.3: FLOWCHART OF MODEM INTERRUPT ROUTINE

NUL	LSB's OF BAUD	MSB's OF BAUD	LSB's OF CLEN	MSB's OF CLEN	LSB's OF FILESIZE	MSB's OF FILESIZE	NUL
-----	------------------	------------------	------------------	------------------	----------------------	----------------------	-----

~~FIG. 5.4~~

BAUD : word variable. storing the encoded value of user defined baud rate for data packet.

CLEN : word variable storing the encoded value of user defined character length and number of stop bits per character for data packets.

FILESIZE: word variable containing the size of data packet that is going to be transmitted.

FIG.- 5.4: STRUCTURE OF CONTROL PACKET

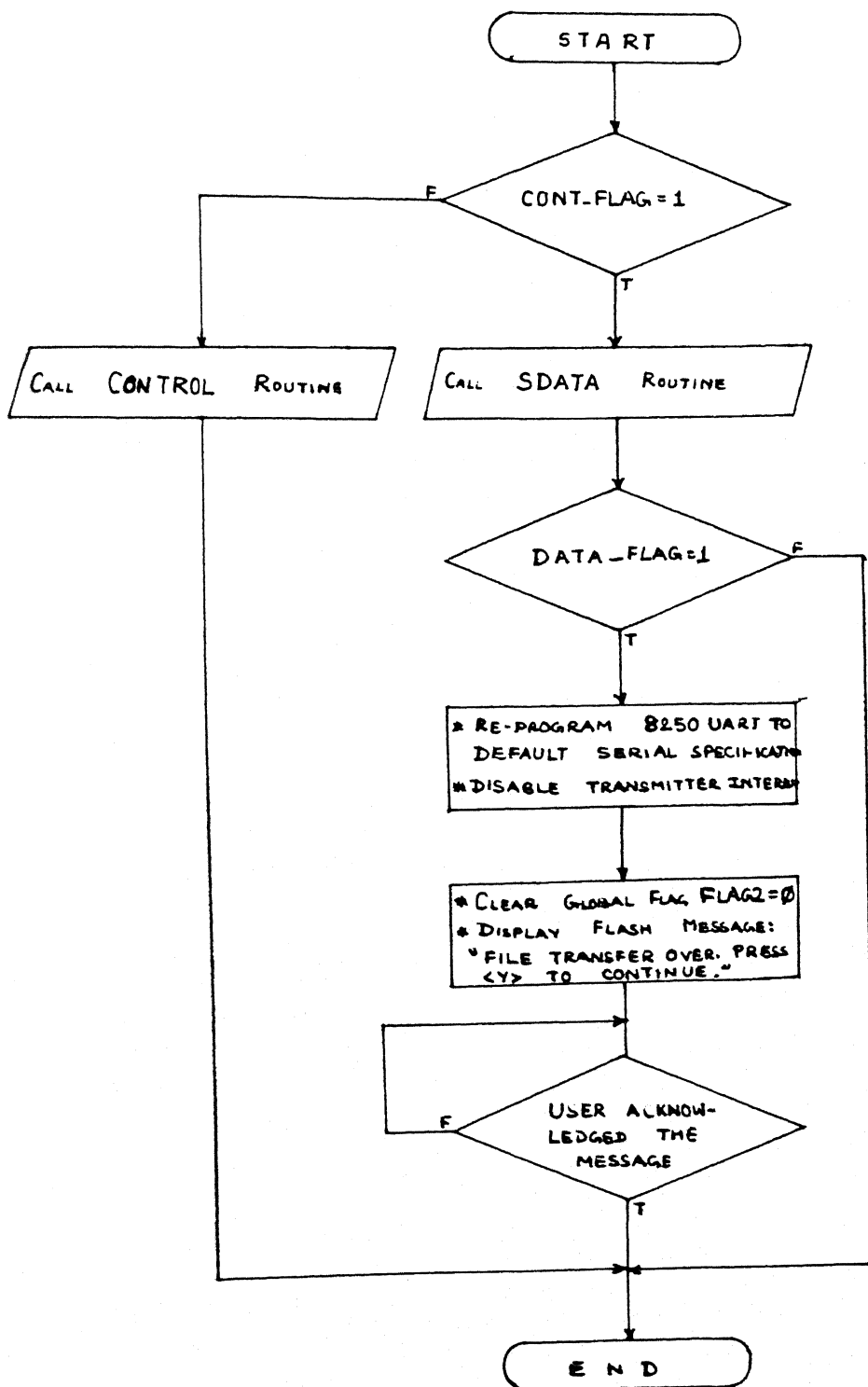


FIG.- 5.5 : FLOWCHART OF TRANSMITTER INTERRUPT ROUTINE

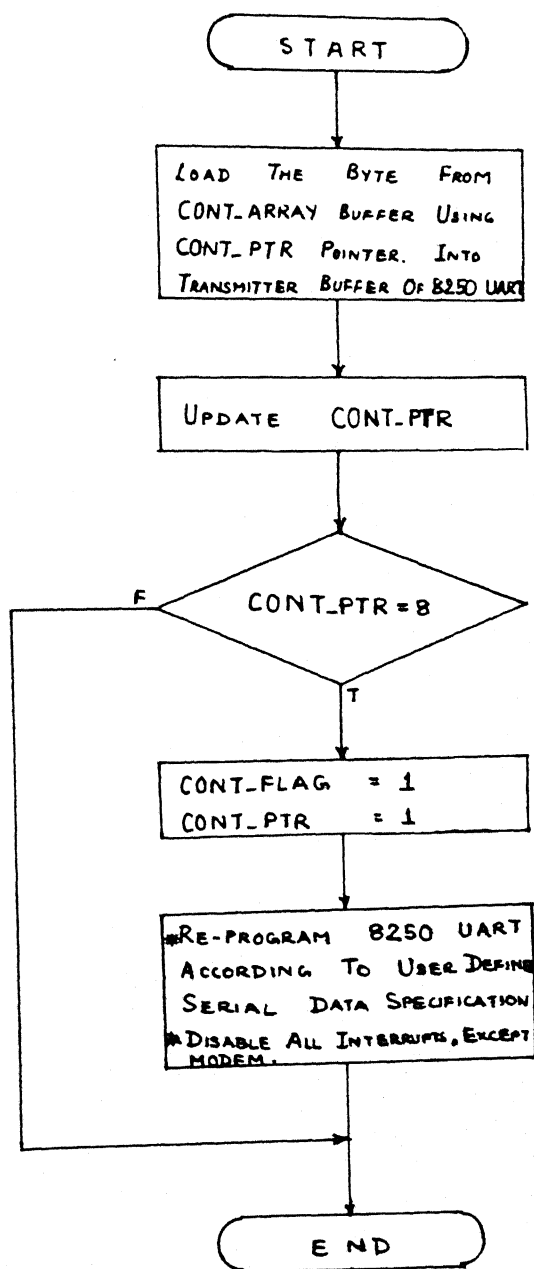


FIG.-5.6: FLOWCHART OF CONTROL ROUTINE

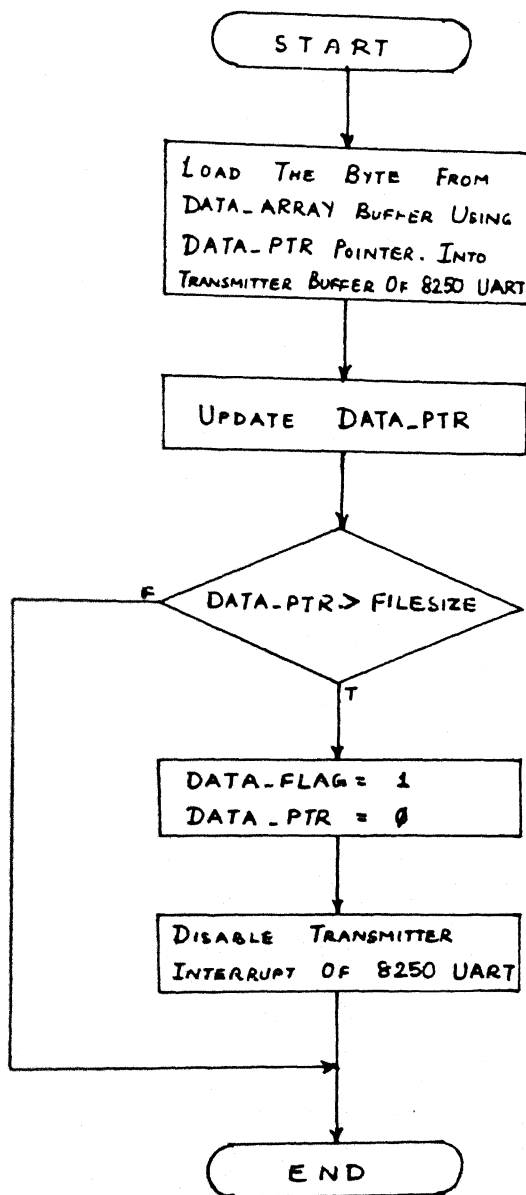


FIG.-5.7: FLOWCHART OF SDATA ROUTINE

5.2.5 Transmitter Interrupt Routine:

There are two set of packets transmitted from secondary node, one is control packet consisting of 8 bytes having header information for central node while the other is data packet containing the data of the file to be printed. The control packet contains the user defined baud rate, character length and stop bits per character at which the data packet is going to be transmitted from the secondary node to the central node. The control packet structure is shown in Fig.5.4. The procedure for transmitting the control packet bytes are different from the transmission of data packet bytes, hence they are discussed separately and the flowchart is given in Fig.5.6 and Fig.5.7.

During the control byte transmission, byte is obtained from the control buffer (CONT_ARRAY) and loaded into transmitter buffer of 8250 UART using a control pointer (CONT_PTR). The CONT_PTR is updated and checked if its value has rolled over 7, if yes, then the control packet transmission is over and CONT_PTR is set to default value 1 and a control flag (CONT_FLAG) is set. Then the COM port is re-programmed according to the user defined baud rate, character length and stop bits per character for data packet transmission. All the interrupts of 8250 UART, except Modem interrupt, are disabled.

During the data packet transmission, the byte is obtained from data buffer (DATA_ARRAY) using data pointer (DATA_PTR) and loaded into transmitter buffer of 8250 UART.

The DATA_PTR is updated and checked to see if it has crossed the FILESIZE variable (stored as a variable when disk is read). If the DATA_PTR has crossed it then the DATA_FLAG is set indicating the end of file transmission. The DATA_PTR is set by default to 0 and finally the transmit interrupt of 8250 UART is disabled.

As shown in Fig.5.5, the CONT_FLAG is checked to see if the control packet has to be sent or data packet has to be sent, accordingly CONTROL or SDATA procedure is executed. If it was data packet transmission, the DATA_FLAG is checked and when this flag is reset to 0, then quit from the transmitter interrupt routine, else re-program the 8250 UART to default serial I/O specification so that it receives the MESSAGE packet from the central node. The transmitter interrupt is disabled. The user is informed about the successful transfer of the file from the secondary node to the central node by displaying a flash message on the bottom of the screen. The user has to acknowledge the message by pressing <Y> key. The global flag FLAG2 will be reset to 0 indicating the print process is over from the secondary node and so the print queue is empty and can accept any other file that has to be printed.

5.2.6 Receiver Interrupt Routine:

The central node times-out after receiving the data packet from the secondary node and acknowledges the secondary node by sending a MESSAGE packet. The message packet contains information about the number of 'overrun',

'framing', 'parity' and 'break detect' errors occurred during the data packet transmission. The message packet also contains the number of bytes received during the data packet transmission, hence if it is less than the FILESIZE variable, it indicates partial successful transmission.

On receiver interrupt of 8250 UART, the receiver buffer of 8250 UART is read and the byte is stored in the message buffer (REC_ARRAY) using receiver pointer (REC_PTR). The REC_PTR is updated and checked if its value has rolled over 7. If not, the Receiver Interrupt Routine is quit, else, REC_PTR is set to 0 and a message is flashed indicating the user about the information received regarding the data packet sent for printing. The user acknowledges the message by pressing <Y>.

5.2.7 'Error on Reception' Interrupt Routine:

During the MESSAGE packet reception, error can occur. On error this interrupt will be invoked. The status register of 8250 UART is read and tested to find the type of the error. The variables regarding the types of errors are updated and can be used by some Network Monitoring program to keep track of errors occurring during reception between any two system initialization moments.

5.3 SOFTWARE FOR CENTRAL NODE:

5.3.1 Protocol Consideration:

The Print Server Controller will sequentially check the serial ports to see if the -RTS lines are asserted by any of the secondary node. Each serial port is allotted a time slot

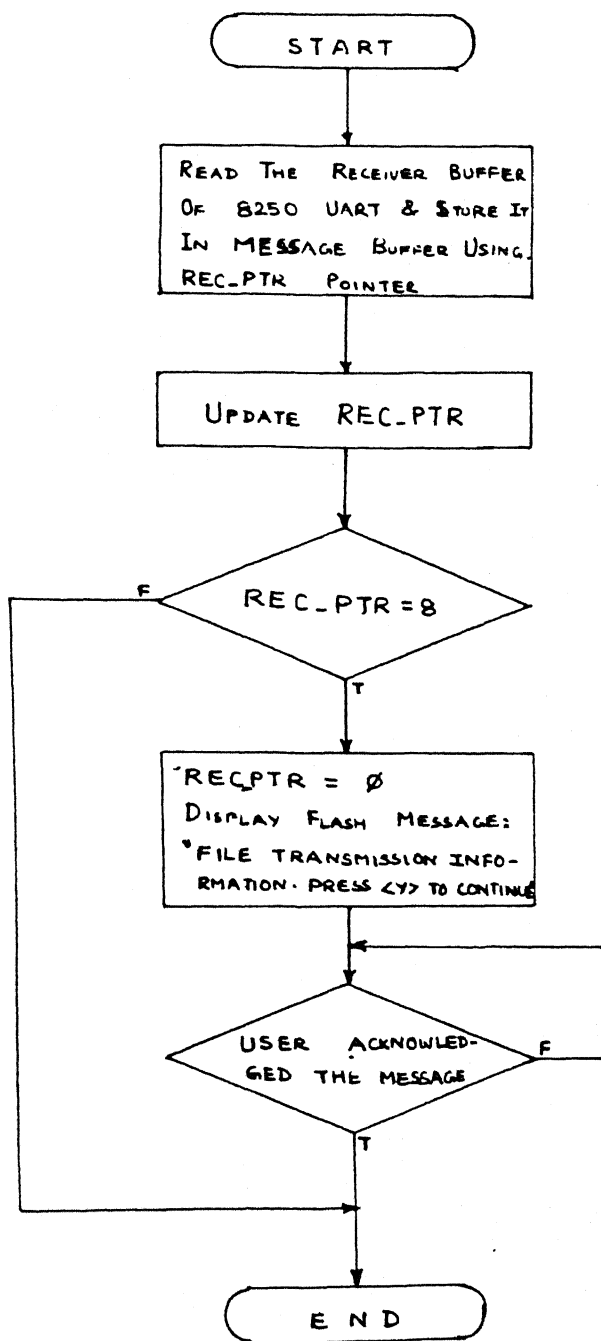


FIG.- 5.8: FLOWCHART OF RECEIVER INTERRUPT ROUTINE.

NUMBER OF OVERRUN ERRORS	NUMBER OF FRAMING ERRORS	NUMBER OF PARITY ERRORS	NUMBER OF BREAK DETECT	TOTAL NUMBER OF ERRORS	LSB OF SIZE OF FILE RECEIVED	MSB OF SIZE OF FILE RECEIVED	NUL
--------------------------------	--------------------------------	-------------------------------	------------------------------	------------------------------	------------------------------------	------------------------------------	-----

FIG.- 5.9: STRUCTURE OF MESSAGE PACKET.

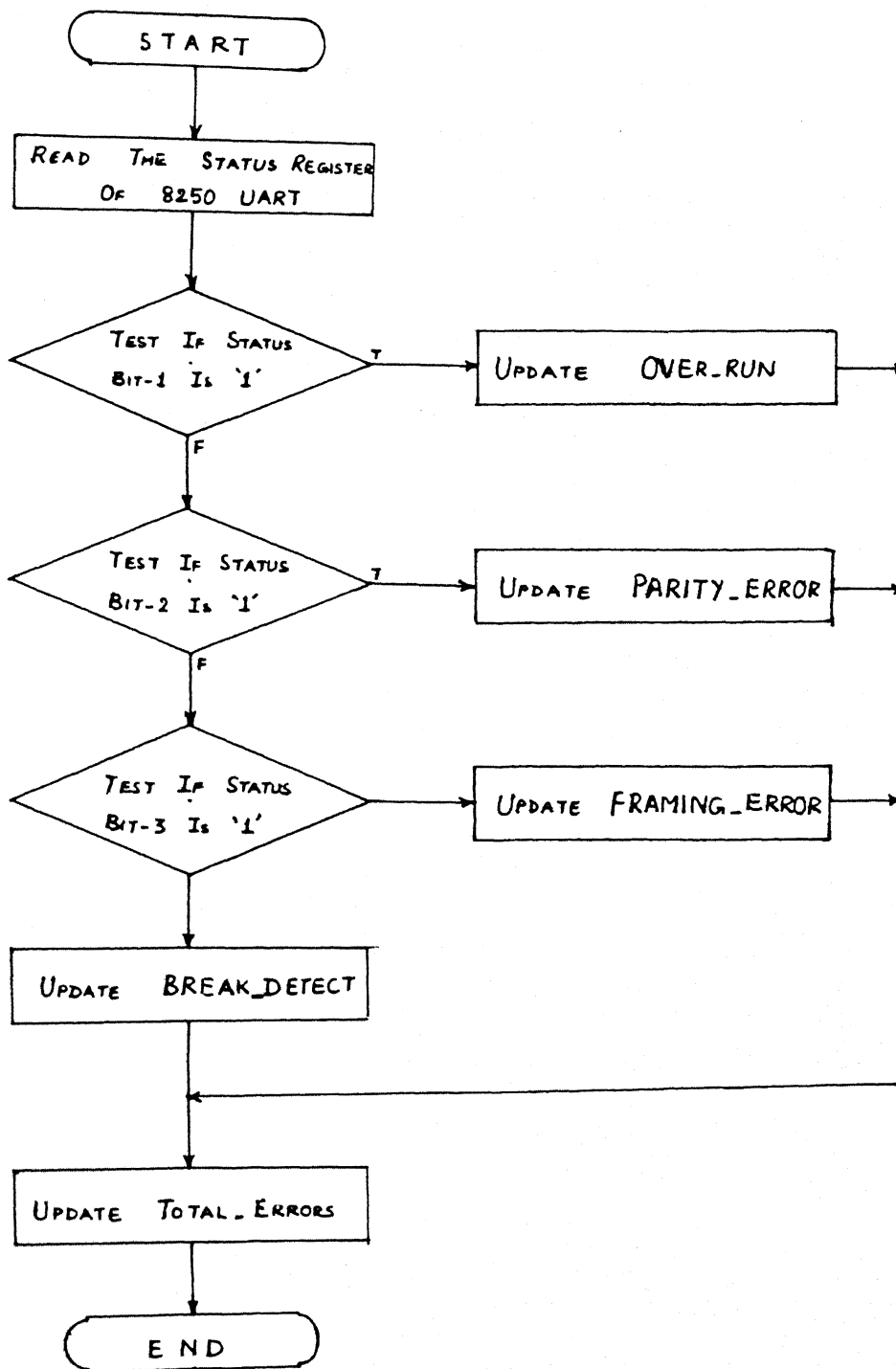


FIG.- 5-10: FLOWCHART OF ERROR DURING RECEPTION INTERRUPT ROUTINE

of a certain maximum duration called 'slots'. This time slot is divided into minislots of particular duration. The controller will sense the -RTS line at the end of each minislot. If -RTS line is in active level, then the controller will go for a short vacation (debounce time) and comes back to check again if the same -RTS line is active. If yes, the controller concludes that the request from the secondary node is a valid one, else it means that the request was spurious one occurring due to some noise on the serial link. If at the end of the minislot the -RTS was sensed inactive then minislot is updated. Once the slot is completed, then the controller will begin scanning of the next serial port. Thus scanning process goes on.

If the controller receives a valid request from the secondary node then it sends a -CTS signal to secondary node to begin the transmission. The receiver of 8256 MUART will receive the control packet at default serial interface specification. The control packet is decoded and re-programs the receiver of 8256 MUART into the user defined specification to receive the data packet. It stores the data byte into DATA_ARRAY and invokes the printer to begin the printing. If the controller encounters any error in byte reception then it updates the error statistics accordingly. If the controller doesn't receive a serial character for a time interval of 16.384 sec., then it assumes the transmission is over. It then transmits a MESSAGE packet to the secondary node and resumes scanning operation.

5.3.2 MAIN_MODULE Routine:

The MAIN_MODULE Routine of central node is very similar to MAIN_MODULE1 Routine of secondary node. This routine is executed in AUTOEXEC.BAT batch file. A global flag FLAG3 is checked to see if it is set (OFFFFH). If the flag is set, then the resident portion of program is already existing in the memory and so quits from the program. If not, then the global flag is set and vector table of IRQ-2 is initialized pointing to the ISR provided in the server software. Then the program is made resident using KEEP function of MS-DOS. The resident portion of the program requires 64Kb memory and quits from the program with successful message.

5.3.3 Scan Interrupt Routine:

During the system initialization, the user~~4~~-0 is given the ownership of the time slot and the number of minislots used during that scanning cycle is set to 0. When the scan interrupts occurs, the owner of the slot is determined by reading the select lines of output pins of Port-1 of 8256 MUART. The -RTS line of that user channel is checked to see if it is asserted. If the user has not asserted the -RTS line, then the minislot variable (number of minislots used) is updated. If a certain maximum number of minislots for that channel is over (10 minislots per scan cycle is allotted to each user, each of 10 mSec duration) then the owner of the slot is changed to the next user in rotation and its minislot variable is reset to 0. If all the minislots for the user is not over then only the minislot

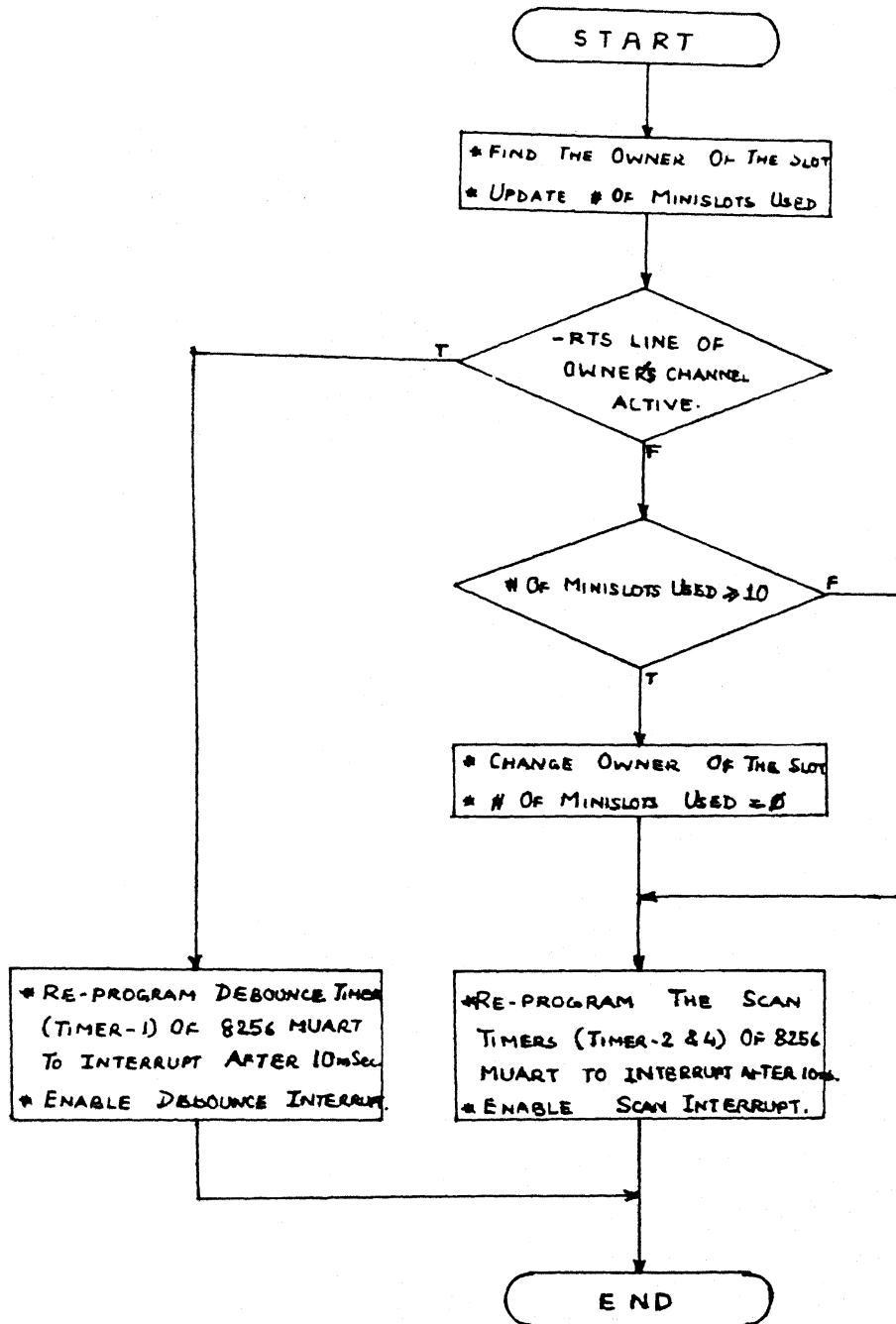


FIG.- 5.11: FLOWCHART FOR SCAN INTERRUPT ROUTINE

variable is updated. The scan timers (Timer-2 and 4 of 8256 MUART) are re-programmed to interrupt again after 10 mSec.

If the user asserts the -RTS line, then it can be a valid or spurious request. To determine the validity of the request for transmitting the file, the controller will go for a vacation period (debounce time) and user has to keep the request active till then to be accepted as a valid request. The debounce timer (Timer-1) is programmed to interrupt after 10 mSec. The interrupt for debounce timer is enabled in 'interrupt set register' of 8256 MUART.

5.3.4 Debounce Interrupt Routine:

The debounce routine is meant to reject a spurious request for transmission. The controller checks the -RTS line again to see if the secondary node has still held it active. If yes, the request is valid and the user is given the right to transfer the file over the transmission channel by acknowledging the request by making -CTS active. The receiver of 8256 MUART is enabled and programmed to accept serial character in default specification. The receiver interrupt and Time-out interrupts are enabled. The time-out is useful so that the central node is not stuck if suddenly the link on which the transmission is occurring fails. The time-out timers (Timer-3 and 5) are programmed to time-out after 16.384 sec. if no byte is received.

If the -RTS line does not stay asserted then it is a spurious interrupt and hence only the minislot variable of the owner of the slot is updated and scan cycle is resumed.

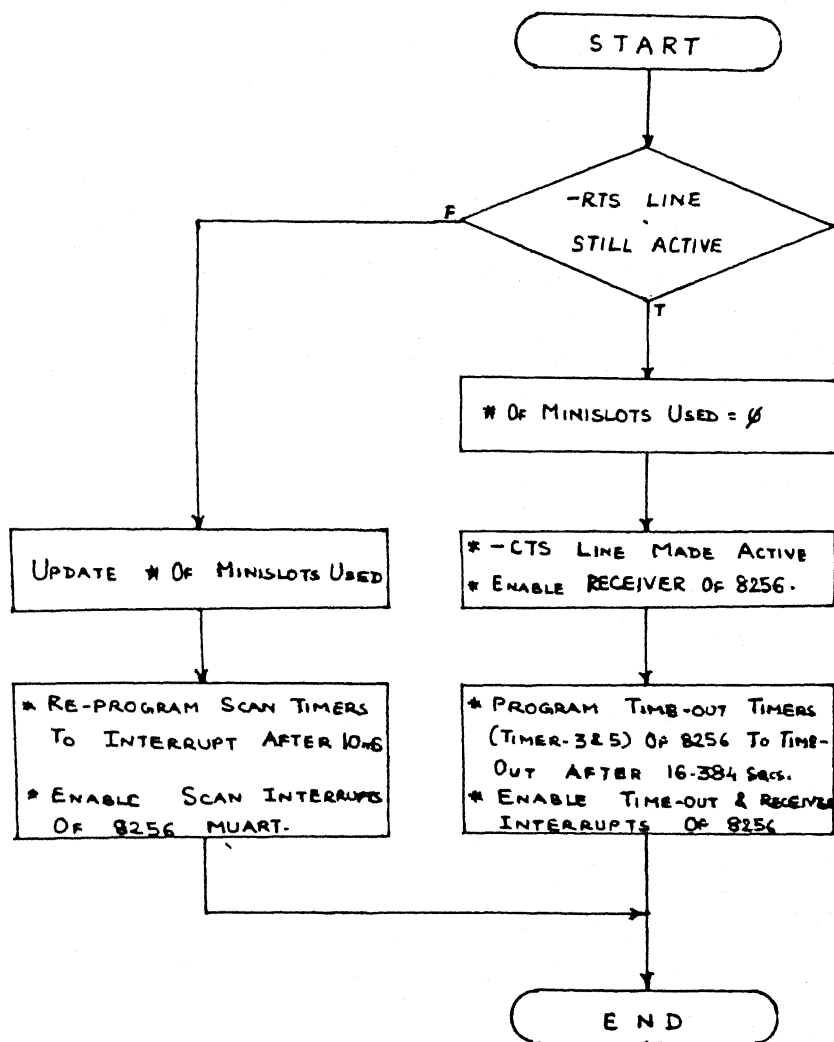


FIG.- 5.12: FLOWCHART FOR DEBOUNCE INTERRUPT ROUTINE.

The scan timers are re-programmed to interrupt after every 10m Sec.

5.3.5 Receiver Interrupt Routine:

Whenever a byte is received from the secondary node, this interrupt is invoked. The status register of 8256 MUART is read to determine if the received byte is in error or not.

If the byte read is in error then the error statistics about the type and number of errors is updated which would be finally sent to the secondary node in a MESSAGE packet at the end of file reception. The central node receives control packet of fixed size and data packet of variable size. The correctness of the data reception for control packet is highly desirable. The error statistics which is compiled for each file transmission is meant only for the data packets. Only four types of errors are recognized, they are Overrun error, Framing error, Parity error and Break Detect error. The byte which is in error is lost for ever and no dummy character is inserted into the buffer meant for storing the received bytes.

If the bytes are received correctly, then it can be of control or data packet, determined by the control flag (CONT_FLAG1). If the CONT_FLAG1 is reset (0), then it is a control packet byte, else byte of data packet. The central node keeps the control packet information of the last 32 files that are transferred to central node from secondary node, this could be useful for network monitoring, etc. Each

control packet is of 8 bytes, the first byte will be 0 followed by 7 bytes containing the serial data format information and the number of bytes that are going to be transferred in the data packet. The control bytes are read from the receiver buffer of 8256 MUART and stored in control array (CONT_ARRAY1) using control pointer (CONT_PTR1). The CONT_PTR1 is updated and checked to see if its value becomes a factor of 8. If yes, then the 8 bytes of control packet are received and the variable associated with data packet transfer like number of data bytes received, number of errors, etc. are reset. The central node makes the -CTS line inactive to the requested secondary node to inhibit the secondary node from flooding the central node with the data bytes till it is ready for data packet transfer. The CONT_FLAG1 is set (1) to indicate that the bytes henceforth received will be of data packet. The information from the control packet is extracted regarding the user defined serial data transfer format and programs the receiver of 8256 MUART accordingly. The central node will make the last byte of control packet contain the information about the owner of the file by reading the select outputs (S0 - S3) of output pins of Port-1 of 8256 MUART. The 8256 MUART is ready to receive data packet bytes and hence sends -CTS line active to enable the data packet transmission.

As the CONT_FLAG1 is set, the byte received is stored in data buffer (DATA_ARRAY1) using data pointer. The data buffer management is explained in another section.

The time-out timers (Timer-3 and 5) are re-programmed so that it can time-out if no more byte is received for next 16.384 sec. The time-out interrupt is enabled.

5.3.6 Time-out Interrupt Routine:

On time-out interrupt occurring the central node concludes that the file transmission from secondary node is over and prepares the MESSAGE packet which will be sent to the secondary node containing the information of the file transfer. The problem can arise if the fore-ground task at the secondary node blocks the COM port interrupts and the data transmission stops for time greater than 16.384 sec., then the central node assumes that the data transfer is over and will send the MESSAGE packet.

The CONT_FLAG1 is reset to 0, so the next data reception will be of control packet. The central node sends -CTS signal inactive to prevent the secondary node from sending any more data. The transmitter of 8256 MUART is programmed to the default specification. The transmitter interrupt is enabled and a dummy character is sent to it.

5.3.7 Transmitter Interrupt Routine:

The central node transmits the MESSAGE packet of 8 bytes. This packet contains acknowledgment for the data packet transfer.

The status register of 8256 MUART is read to determine if transmitter is the cause of the interrupt. The byte is read from message buffer using message pointer (MES_PTR) and loaded into the transmitter buffer of 8256 MUART.

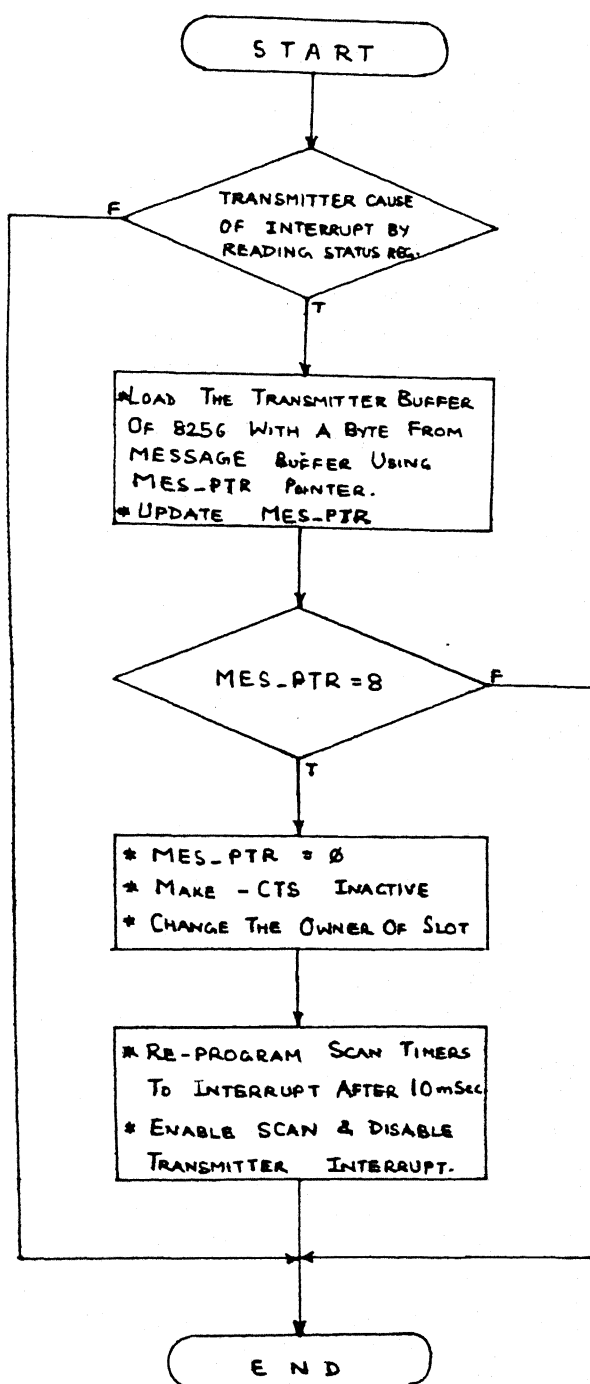


FIG.- 5.13: FLOWCHART FOR TRANSMITTER INTERRUPT ROUTINE (CENTRAL NODE).

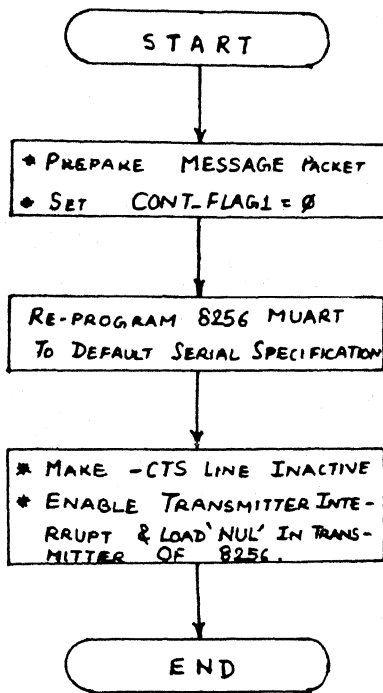


FIG.- 5.14: FLOW CHART FOR TIME-OUT INTERRUPT ROUTINE

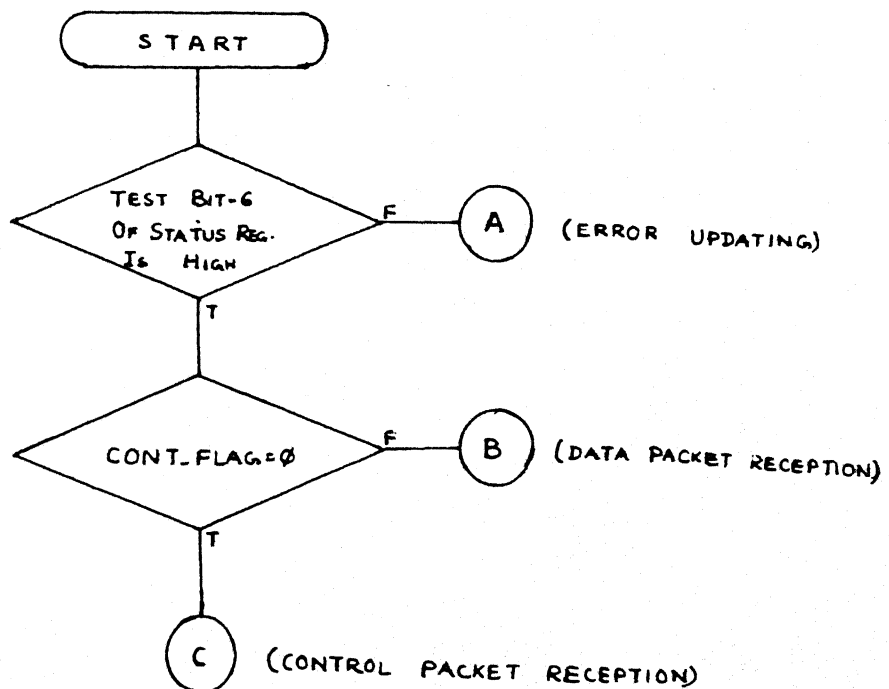


FIG.- 5.15: FLOWCHART FOR- RECEIVER INTERRUPT ROUTINE
(CENTRAL NODE)

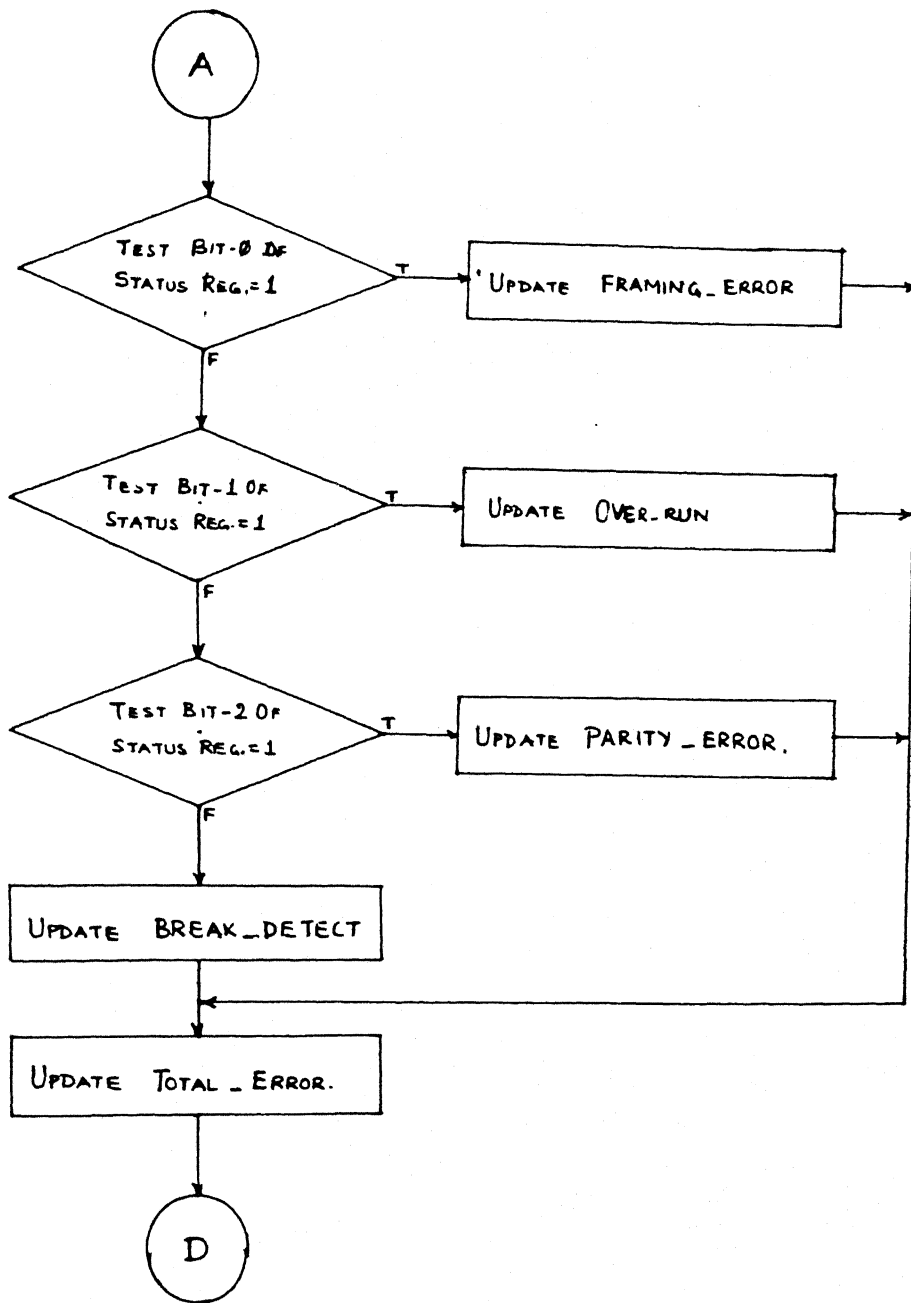
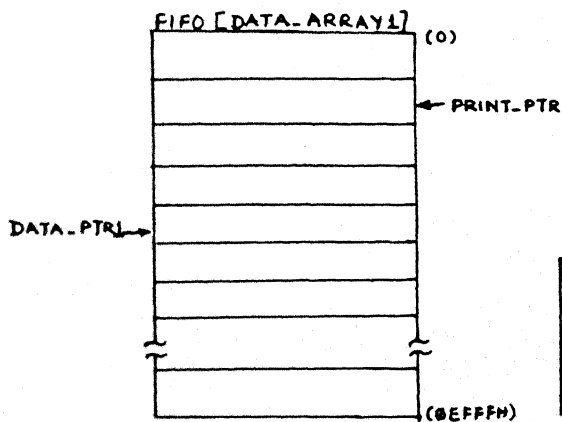


FIG.- 5.16 : FLOWCHART FOR ERROR UPDATING.



BUFFER_STAT	SERIAL INPUT	PARALLEL OUTPUT
EMPTY	ON	OFF
INUSE	ON	ON
FULL	OFF	ON

FIG.- 5.17: FIFO STRUCTURE & STATUS

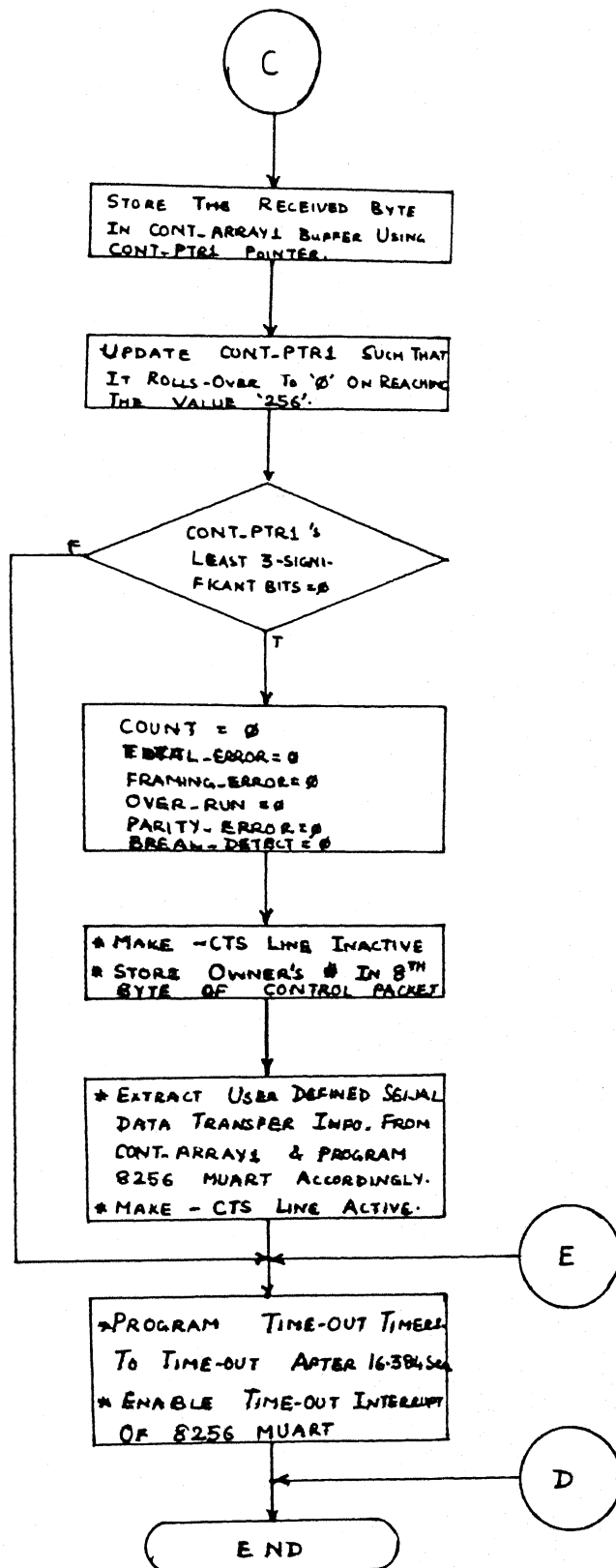


FIG.-5.18: FLOWCHART FOR CONTROL PACKET RECEPTION.

The MES_PTR is updated and is checked if it rolls over the value 8. If yes, the message packet transmission is over. The transmitter interrupt and time-out interrupt are disabled and the ownership of the slot is updated to the next user in rotation. The scan timers are programmed to interrupt after 10 mSec, thus scanning cycle gets re-started.

5.3.8 Printer Routine:

The card has the capability of being interfaced to DATAPRODUCTS and CENTRONICS Interface Line Printers. Here the software has been developed for only one Centronics Interface Line Printer, so software developed for it will be discussed.

The byte is removed from DATA_ARRAY1 using printer pointer (PRINT_PTR) and loaded into Port-A of 8255 PPI. The STROBE (bit-7 of Port-C of 8255 PPI) pin is set low for nearly 2uS duration and then again reset to high level using bit set/reset function of 8255 PPI. The STROBE pulse will clear the flip-flop latching the -ACK pulse. This signal also acts as a STROBE to the printer to accept the data available on Port-A pins of 8255 PPI. The buffer management is described separately.

5.3.9 Buffer Management:

The FIFO implementation uses a 60Kb array to store the characters. There are two pointers used as indices in the array to address the characters: DATA_PTR1 and PRINT_PTR. The DATA_PTR1 points to the location in the array which will store the next byte of the data inserted, while PRINT_PTR

points to the next byte of data which will be removed from the array for printing. Both the variables are declared as word variables. The FIFO is illustrated in block diagram as shown in Fig.5.17.

The bytes of data packets are received from Receiver Interrupt Routine and stored in the array location pointed to by DATA_PTR1. The DATA_PTR1 is incremented. Similarly the byte is removed from the array by Printer Routine using PRINT_PTR and then PRINT_PTR is incremented. Since both the pointers are only incremented, they must roll-over when they hit the top of 60Kb address space. The DATA_PTR1 and PRINT_PTR also indicate how many bytes are in the FIFO and whether the FIFO is full or empty. When a character is placed into the FIFO and DATA_PTR1 is incremented, the FIFO becomes FULL if DATA_PTR1 catches up with PRINT_PTR. When a character is read from the FIFO and PRINT_PTR is incremented, the FIFO becomes EMPTY if PRINT_PTR catches with DATA_PTR1. If the FIFO is neither FULL nor EMPTY, then it must be INUSE. A variable called BUFFERSTAT, is used to indicate one of the three status condition of FIFO. The software uses the BUFFERSTAT to control the flow into or out of the FIFO. When the FIFO is EMPTY, the printer is disabled from putting any request for data to print. When the FIFO is FULL, -CTS must be made inactive so that no more data byte will come and cause a buffer overflow. If the BUFFERSTAT is INUSE status, then -CTS line is active and printer request stays enabled.

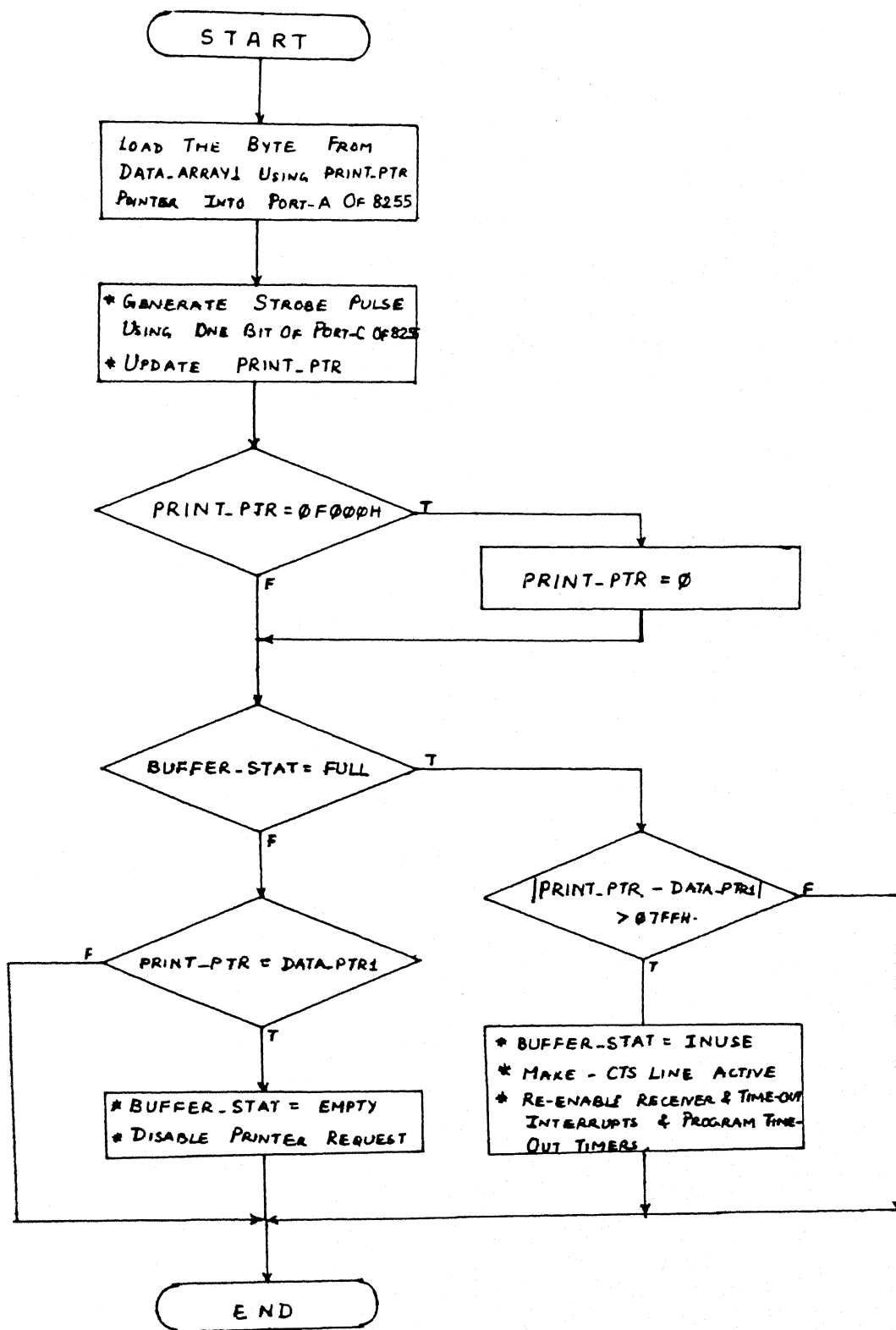


FIG.- 5.20: FLOWCHART FOR PRINTER ROUTINE

The flowchart of data packet reception part of Receiver Interrupt Routine is shown in Fig.5.19. The received byte is stored in the FIFO (DATA_ARRAY1) using DATA_PTR1 and DATA_PTR1 is updated and checked to see if it has hit the top of 60Kb address space. If yes, then DATA_PTR1 is reset to 0. The BUFFERSTAT is checked to see if it is EMPTY. If yes, then a NUL character is dumped into Port-A of 8255 PPI and a STROBE pulse is generated and BUFFERSTAT is set to INUSE condition. If the BUFFERSTAT wasn't EMPTY, then DATA_PTR1 is checked to see if it equals PRINT_PTR. If both the pointer becomes equal, the BUFFERSTAT is set to FULL condition and -CTS line is made inactive to inhibit the secondary node from transmitting for time being. The time-out timers are disabled to prevent false time-out alarms occurring, when secondary is prevented from sending any byte.

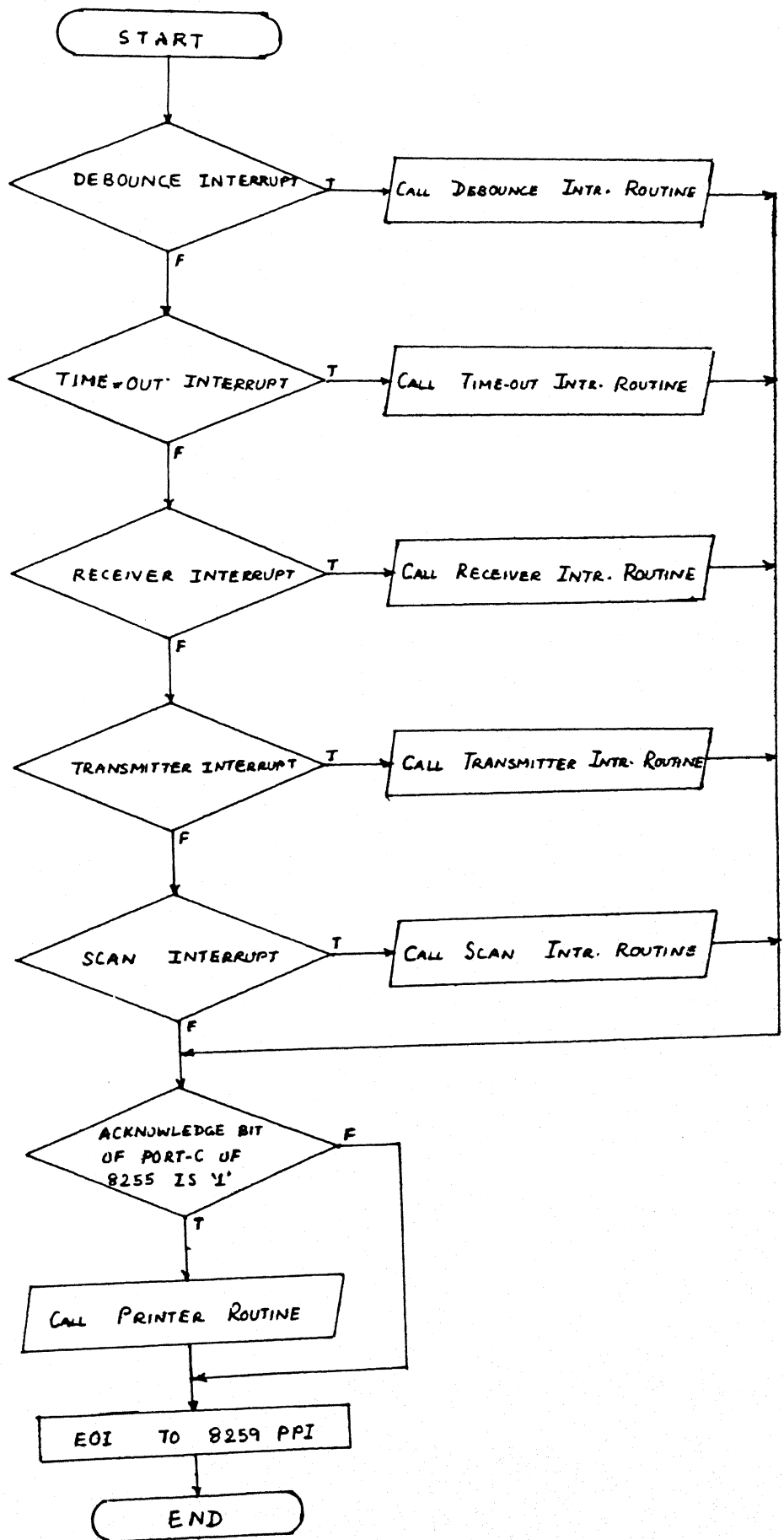
The flowchart of the Printer Routine is shown in Fig.5.20. After the byte is read from the FIFO and loaded into Port-A of 8255 PPI, the PRINT_PTR is updated and checked to see if it has hit the top of 60KB of address space. The PRINT_PTR is rolled over if need be. The BUFFERSTAT is tested to determine if it is FULL,. If yes, then it is checked to see if atleast 2KB buffer space is available in the FIFO. If this is also true, then the serial data reception is re-enabled by making -CTS line active and so buffer can again fill up. The BUFFERSTAT is set to INUSE state, the time-out timers are programmed to time-out after

16.384 sec., Receiver interrupts and Time-out timer interrupts are enabled.

If BUFFERSTAT was tested to be not FULL, then PRINT_PTR is checked to see if it has become equal to DATA_PTR1. If yes, the BUFFERSTAT is set to EMPTY and printer request is disabled by making STROBE line continuously low, so that the Flip-Flop used for latching the -ACK pulse from the printer stays clear.

5.3.10 Check Routine:

On interrupt request on IRQ2 line, this routine is invoked. The interrupt address register is read by CPU to acknowledge the interrupt request of 8256 MUART. The cause of the interrupt is also known by reading the interrupt address register. As shown in Fig.5.21 the interrupt address register is checked to see if Receiver, Transmitter, Time-out Timer, Scan Timer or Debounce Timer is the cause of the interrupt. After servicing the 8256 MUART interrupt, the Port-C of 8255 PPI is read to see if ACKNOWLEDGE bit is set. If yes, the printer routine is executed. Thus the printer is serviced in a polled mode. In the end End Of Interrupt (EOI) command is given to 8259 PIC to be able to service later interrupts.



CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 CONCLUSION:

The present work is intended to design and implement a Print Server on IBM PC/XT. The Printer support like CENTRONICS and DATAPRODUCTS Interface for Line Printers and RS-232C Interface for Serial Printers are implemented and tested by simulation. The software support for CENTRONICS Interface Printer was implemented using L & T Dot Matrix Printer. The testing was done with only one secondary node attached to the central node, but the result stays valid till 16 such nodes are connected. The files of size less than 58kB was transferred properly at baud rate ranging from 300 to 9600 baud. The Print Server software at central node and secondary node were running as the background job and users were able to do their normal editing work etc. in the fore-ground mode. The present work lacks a user friendly menu.

6.2 RECOMMENDATIONS:

6.2.1 Hardware Level:

To improve upon the response time for the request from the secondary node, separate processor can be used with RAM and EPROM so that the 8088 CPU of IBM PC/XT is not overloaded. This scheme is costlier in comparison to

to the present scheme. The address-data multiplexing logic will not be required because the 8088 processor's address-data bus could be directly connected to 8256 MUART. The RAM can be of size 128kB to 256kB to provide for the file size of length greater than 58kB but should be less than the RAM size to be printed. The card will communicate with PC through I/O Channel using interrupt lines when it wants to use PC resources like display, keyboard etc. Then the first and second card of the present version can be unified into one card and housed in a separate box and connected to PC through a flat ribbon cable and communicating through I/O Channel of PC.

6.2.2 Software Level:

The software for DATAPRODUCTS Interface Printer and Serial Printer can be written and the user can be given the option of different printers. The menu for Print Server could be made more user friendly and other features like displaying the print queue status of the Central node, deleting the print request from the print queue, providing priority for superuser. The central node and secondary nodes are connected in a Star Topology with full duplex communication channel available between the central and the secondary nodes, this can be utilized to provide for file transfer facility from one secondary node to another secondary node via central node.

BIBLIOGRAPHY

- [1] James L. Peterson and Abraham Silberscham, "Operating Systems Concepts", Addison Wesley Publishing Company Inc., 1983.
- [2] Ray Duncan, "Advanced MSDOS", Microsoft Press, 1986.
- [3] Yu-Cheng Liu & Glenn A. Gibson, "Microcomputer Systems: The 8086/8088 Family (Architecture, Programming and Design)", Prentice Hall India, 1986.
- [4] Alan R. Miller, "Assembly Language Techniques for the IBM PC", BPB Publications, 1987.
- [5] Michael L. Gurrie & Patrick J. O'Connor, "Voice/Data Telecommunication System : An Introduction to Technology", Prentice Hall Inc., 1986.
- [6] Cay Weitzman, "Distributed Micro/Minicomputer Systems - Structure, Implementation & Application", Prentice Hall Inc., 1980.
- [7] Uyles Black, "Computer Networks: Protocols, Standards, and Interfaces", Prentice Hall Inc., 1987.
- [8] "INTEL : Microsystem Components Handbook - Peripherals Volume-II", pp.6-351 to 6-373 and 6-386 to 6-460, Intel Corporation, 1986.
- [9] "Technical Reference - Personal Computer XT Systems", IBM Release, Part No.6936832.
- [10] "L&T Dot Matrix Printers User's Manual", L&T Limited.

APPENDIX-A

CALCULATION OF AVERAGE WORST CASE DELAY FOR A RESPONSE FROM CENTRAL NODE

As shown in Fig.1, if user#1 transmits a file, the time taken by that node using the central node's communication channel is:

$$t_{1,1} = n_{1,1} \cdot \delta + \delta + X_{1,1} + A + B \quad (\text{eqn. \#1})$$

(first term on right side is Scanning time, next term is Debounce time, next is data receive time, next is Time-out of reception time and last term is message packet transmission time.)

where δ = minislots interval.

$n_{1,1}$ = number of minislots required till -RTS is sensed active during scanning phase.

$1 \leq n_{1,1} \leq N$.

N = maximum number of minislots permitted to each user in a scanning phase.

$X_{1,1}$ = time to receive a file from sending node. (which include the time buffer is full and transmission has to be stopped.)

A = constant reception time-out time required for concluding the reception is over.

B = constant time required for transmitting the MESSAGE packet to secondary node.

Similarly for transmitting File#2 from same user, the time used on communication channel is

$$t_{1,2} = n_{1,2} \cdot \delta + \delta + X_{1,2} + A + B \quad (\text{eqn. \#2})$$

Now a constant term $Z = A + B + \delta$

The time used of communication channel when file#k is sent by user#1 is:

$$t_{1,k} = n_{1,k} \cdot \delta + X_{1,k} + Z \quad (\text{eqn. \#3})$$

The average time used by user#1 if it is sending file :

$$t_1 = n_1 \cdot \delta + X_1 + Z \quad (\text{eqn. \#4})$$

where n_1 and X_1 are the mean number of minislots required before -RTS is sensed active and average time for file reception.

If the user is not sending a file, the central node will scan its -RTS line for the whole scan duration. In that case the time used by the user is:

$$t_1' = N \cdot \delta \quad (\text{eqn. \#5})$$

Now for user#1, the duty cycle (number of file transmitted per scanning cycle) with which a file is transmitted is ' f_1 ' ($0 \leq f_1 \leq 1$). As a result the average time the user#1 is using the central node's communication link :

$$T_1 = f_1 \cdot t_1 + (1 - f_1) \cdot t_1' \quad (\text{eqn. \#6})$$

Using (eqn. \#4) and (eqn. \#5) in (eqn. \#6)

$$T_1 = f_1 \cdot [n_1 \cdot \delta + X_1 + Z] + (1 - f_1) \cdot [N \cdot \delta] \quad (\text{eqn. \#7})$$

$$= N \cdot \delta + f_1 \cdot [n_1 \cdot \delta + X_1 + Z - N \cdot \delta]$$

$$= N \cdot \delta + f_1 \cdot n_1 \cdot \delta + f_1 \cdot X_1 + f_1 \cdot Z' \quad (\text{eqn. \#8})$$

where $Z' = Z - N \cdot \delta$

Similarly, the average time used by the user#2 is:

$$T_2 = N \cdot \delta + f_2 \cdot n_2 \cdot \delta + f_2 \cdot X_2 + f_2 \cdot Z' \quad (\text{eqn. \#9})$$

Hence the average time used by any user is:

$$T = (T_1 + T_2 + \dots + T_m) / m$$

where ' m ' = number of user node in the system.

$$T = N \cdot \delta + \delta \cdot [(\sum f_i \cdot n_i) / m] + (\sum f_i \cdot X_i) / m + Z' \cdot [(\sum f_i) / m]$$

Now $F = (\sum f_i * n_i) / m$ = mean number of minislots required by users during scanning cycle.

$G = (\sum f_i * X_i) / m$ = mean time required for a file to be get transmitted from secondary node to central node.

$f = (\sum f_i) / m$ = average duty cycle of the system.

$$T = N * \delta + F * \delta + G + (Z - N * \delta) * f \quad (\text{eqn. \#10})$$

$$= (F * \delta + G + Z * f) + (1 - N * \delta) * f \quad (\text{eqn. \#11})$$

(the first term is the communication channel time usefully utilized and second term is the channel time going as waste.)

The average worst case delay ' T_d ' for any response is when -RTS line is raised slightly after the scanning slot of that user is over and the scanning slot of next user begins.

$$T_d = (m - 1) * T \quad (\text{eqn. \#12})$$

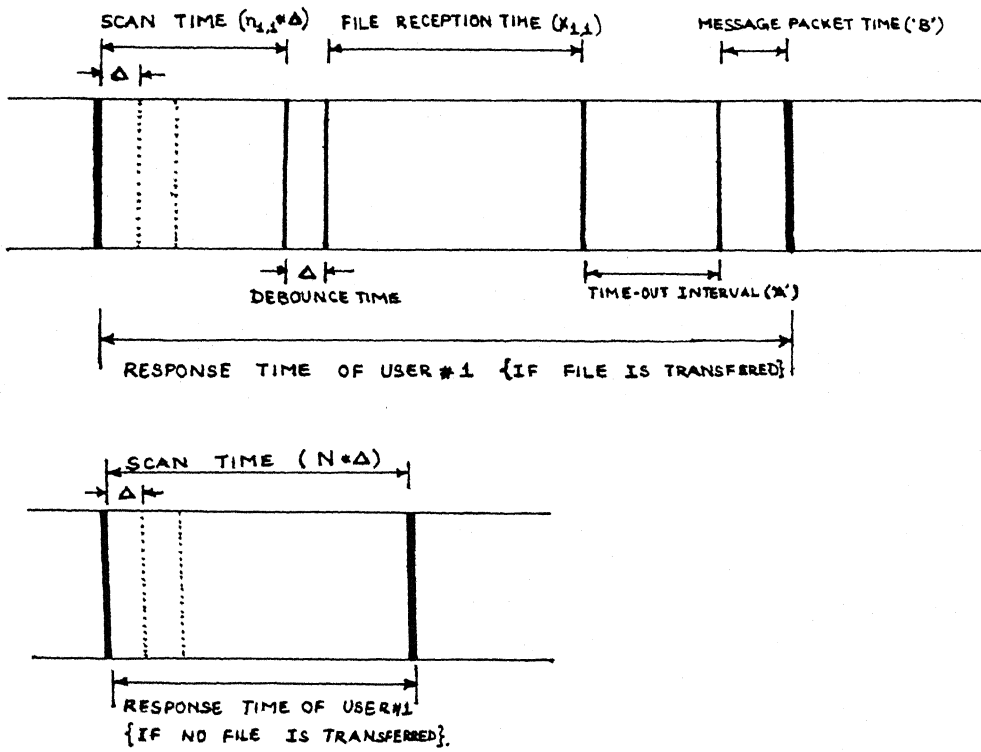


FIG-1: RESPONSE TIME OF A USER.

APPENDIX B : SOFTWARE LISTING

```
#include <stdio.h>
#include <dos.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys\stat.h>

int d,i=0,j=0,row=0,col=1,page=1,fm=2,hm=2,mb=8,mt=3,pl=66,text=55;
int bidir=0,sefb=0,sefd=0,sefu=0,sett=0,sefv=0;
char ibuffer[2048],obuffer[3000];
char name[80],name1[80],ht[40]={0x1b,0x44,0},vt[40],fo[60]={0},he[60]={0};
int hand1=-1,hand2=-1,byte1=-1,byte2=-1,size1=2048,size2=3000,hand3,hand4;
long fp1=0,fp2=0;

/* routine for escape sequence look-up table*/
conversion()
{
    switch(d)
    {
        case 0:horizon();break;
        case 1:obuffer[j]=0x1b;++j;obuffer[j]=0x4d;break;
        case 2:if (sefb==0) {obuffer[j]=0x1b;++j;obuffer[j]=0x45;setb=1;}
            else {obuffer[j]=0x1b;++j;obuffer[j]=0x46;setb=0;}break;
        case 3:--j;break;
        case 4:if (sefd==0) {obuffer[j]=0x1b;++j;obuffer[j]=0x47;setd=1;}
            else {obuffer[j]=0x1b;++j;obuffer[j]=0x48;setd=0;}break;
        case 5:--j;break;
        case 6:obuffer[j]=0x20;break;
        case 7:obuffer[j]=0x7f;break;
        case 8:obuffer[j]=0x08;--col;break;
        case 9:obuffer[j]=0x09;col=1+col+(8-(col%8));break;
        case 10:obuffer[j]=0x0a;++row;break;
        case 11:break;/* special*/
        case 12:obuffer[j]=0x0c;col=1;row=0;++page;break;
        case 13:obuffer[j]=0x0d;col=1;break;
        case 14:obuffer[j]=0x1b;++j;obuffer[j]=0x50;break;
        case 15:--j;break;
        case 17:--j;break;
        case 18:--j;break;
        case 19:if (sefu==0) {obuffer[j]=0x1b;++j;obuffer[j]=0x2d;
            ++j;obuffer[j]=1;setu=1;}
            else {obuffer[j]=0x1b;++j;obuffer[j]=0x2d;
            ++j;obuffer[j]=0;setu=0;}break;
        case 20:if (sett==0) {obuffer[j]=0x1b;++j;obuffer[j]=0x53;
            ++j;obuffer[j]=0;sett=1;}
            else {obuffer[j]=0x1b;++j;obuffer[j]=0x54;sett=0;}break;
        case 22:if (sefv==0) {obuffer[j]=0x1b;++j;obuffer[j]=0x53;
            ++j;obuffer[j]=1;setv=1;}
            else {obuffer[j]=0x1b;++j;obuffer[j]=0x54;setv=0;}break;
        case 23:--j;break;
        case 24:strike();break;/*special*/
        case 25:--j;break;
        case 27:++i;obuffer[j]=ibuffer[i];++i;++col;break;
        case 46:{if (col==1) select();
            else { obuffer[j]=ibuffer[i] & 0x7f;++col;}}break;
        default:obuffer[j]=ibuffer[i] & 0x7f;++col;break;}
    ++i;++j;}

/*move cursor to fixed position*/
/*set printing in ELITE mode*/
/*bold character*/
/*no pause between printing*/
/*double strike character*/
/*custom built not provided*/
/*phantom space as SPACE character*/
/*phantom rubout character*/
/*back space*/
/*horizontal fixed tab*/
/*line feed*/
/*not provided*/
/*form feed*/
/*carriage return*/
/*set printing in PICA mode*/
/*binding space*/
/*custom built not provided*/
/*custom built not provided*/
/*underline character*/
/*superscript character*/
/*subscript character*/
/*custom built not provided*/
/*strike-out words*/
/*italics or color not provided*/
/*graphics character set*/
/*dot command if column no. is 1*/
/*normal character*/

/*Routine for shifting printer carriage to fixed position*/
horizon()
{ obuffer[j]=0x1b;++j;obuffer[j]=0x44;++j;++i;obuffer[j]=ibuffer[i] & 0x7f;
  ++j;obuffer[j]=0;++j;obuffer[j]=0x09;sendht();}

/*Routine for restoring the original horizontal tab position*/
sendht()
```

```

{ int k=0;
  if (ht[k]!=0) { ++j;obuffer[j]=ht[k];++k;}
  else { ++j;obuffer[j]=0;}}

/*Routine for striking out the word*/
strike()
{ ++i;while (ibuffer[i]!=0x24 || 0xa4 ) { ibuffer[i]=ibuffer[i] & 0x7f;
  obuffer[j]=ibuffer[i];++j;obuffer[j]=0x08;++j;obuffer[j]=0x2d;++i;}}

/*Routine for dot commands*/
select()
{ int f,g,k=0;
  ++i;ibuffer[i]=ibuffer[i] & 0x7f;f=ibuffer[i];

  switch(f){ case 66:if (bidir==0) {obuffer[j]=0x1b;++j;obuffer[j]=0x55; /*bidirectional printing*/
    ++j;obuffer[j]=1;bidir=1;}
    else {obuffer[j]=0x1b;++j;obuffer[j]=0x55;
    ++j;obuffer[j]=0;bidir=0;}
    while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a)
      { ++i;ibuffer[i]=ibuffer[i] & 0x7f;}break;

  case 67:++i;if (ibuffer[i]==0x57 || 0xd7) {++i;g=ibuffer[i] & 0x7f;
    if (g==24 || 20 || 17 || 15 || 12 ) /*character width definition*/
      {obuffer[j]=0x1b;++j;obuffer[j]=0x50;}
    else { if (g==10 || 8 )
      {obuffer[j]=0x1b;++j;obuffer[j]=0x4d;}
      else { if ((g==6) ||(g== 5) ||(g== 4 ))
        {obuffer[j]=0x1b;++j;obuffer[j]=0x0f;}}}}
    while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a)
      {++i;ibuffer[i]=ibuffer[i] & 0x7f;}}break;

  case 70:++i;if (ibuffer[i]==0x4d || 0xcd) /*footing margin or footing text*/
    {++i;fm=ibuffer[i] & 0x7f;sendvt();
    while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a)
      {++i;ibuffer[i]=ibuffer[i] & 0x7f;}}
    else if (ibuffer[i]==0x4f || 0xcf)
      {++i;ibuffer[i]=ibuffer[i] & 0x7f;
      while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a)
        {fo[k]=ibuffer[i];++i;+k;
        ibuffer[i]=ibuffer[i] & 0x7f;}
      fo[k]=0;}break;

  case 72:++i;if (ibuffer[i]==0x4d || 0xcd) /*heading margin or heading text*/
    {++i;hm=ibuffer[i] & 0x7f;sendvt();
    while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a)
      {++i;ibuffer[i]=ibuffer[i] & 0x7f;}}
    else if (ibuffer[i]==0x45 || 0xc5)
      {++i;ibuffer[i]=ibuffer[i] & 0x7f;
      while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a)
        {he[k]=ibuffer[i];++i;+k;
        ibuffer[i]=ibuffer[i] & 0x7f;}
      he[k]=0;}break;

  case 73: while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a) /*comments not to be printed*/
    {++i;ibuffer[i]=ibuffer[i] & 0x7f;}break;

  case 46: while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a) /*comments not to be printed*/
    {++i;ibuffer[i]=ibuffer[i] & 0x7f;}break;

  case 76: ++i;if (ibuffer[i]==0x48 || 0xc8) {++i;g=ibuffer[i] & 0x7f;
    obuffer[j]=0x1b;++j;obuffer[j]=0x33;++j;} /*line density per inch*/
    switch(g) { case 24:obuffer[j]=0x6c;break;
      case 20:obuffer[j]=0x54;break;
      case 18:obuffer[j]=0x53;break;
      case 16:obuffer[j]=0x48;break;
      case 12:obuffer[j]=0x36;break;
      case 10:obuffer[j]=0x2d;break;
      case 9:obuffer[j]=0x28;break;
      case 8:obuffer[j]=0x24;break;
      case 7:obuffer[j]=0x20;break;
      case 6:obuffer[j]=0x1b;break;
      case 5:obuffer[j]=0x16;break;

```

```

        default: obuffer[ij]=0x1b; break;}
        while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a)
            {++i; ibuffer[i]=ibuffer[i] & 0x7f; break;}

    case 77: ++i; if (ibuffer[i]==0x42 || 0xc2) /*top or bottom margin*/
        {++i; mb=ibuffer[i] & 0x7f; sendvt();}
        else if (ibuffer[i]==0x54 || 0xd4)
        {++i; mt=ibuffer[i] & 0x7f; sendvt();}
        while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a)
            {++i; ibuffer[i]=ibuffer[i] & 0x7f; break;}

    case 80: ++i; if (ibuffer[i]==0x41 || 0xc1) /*page length or page break*/
        {++i; text=ibuffer[i] & 0x7f; sendvt();}
        else if (ibuffer[i]==0x4c || 0xcc)
        {++i; pl=ibuffer[i] & 0x7f; sendvt();}
        while (ibuffer[i-1]!=0x0d || ibuffer[i]!=0x0a)
            {++i; ibuffer[i]=ibuffer[i] & 0x7f; break;}

    default: break;}}

/*routine for adjustment for vertical tab settings*/
sendvt()
{ int n1,n3,n4,n5;
  n1=mt-hm; n3=pl-mt-mb; n4=mt+n3; n5=n4+fm;
  if (text>n3) text=n3;
  obuffer[ij]=0x1b; ++j; obuffer[ij]=0x42; ++j; obuffer[ij]=n1; ++j;
  obuffer[ij]=mt; ++j; obuffer[ij]=n4; ++j; obuffer[ij]=n5; ++j; obuffer[ij]=0;}

input()
{ while (ibuffer[i]!=0) {d=ibuffer[i]; d=d & 0x7f; printf("\n %x",d); conversion();}
  printf("\n display"); j=0;
  while (obuffer[ij]!=0) {printf("\n %x,%d",obuffer[ij],row); ++j;}
  printf("\n %d,%d,%d,%d,%d,%d",fm,hm,mb,mt,text,pl);} /*

/*Routine for opening a file*/
file_open()
{ printf ("\n file name :");
  scanf ("%s",name);
  hand1= open (name,O_RDONLY);
}

/*Routine for reading a file and loading into a fixed size buffer*/
file_read()
{ byte1= read(hand1,ibuffer,2048);
  if (byte1 <=0) { if (byte1==0) size1=byte1;
    else printf("\n error while reading");}
  else size1=byte1;
}

/*Routine for creating a new file for writing*/
file_creat()
{ int ch,t=0;
  ch=name[t];
  while(ch!='.') {if (ch!=0) {if (t<=77) {name1[t++]=ch; ch=name[t];}
    else break;} else break;}
  name1[t]='.'; name1[t+1]='p'; name1[t+2]=0;
  hand2= creat(name1,S_IREAD | S_IWRITE);
}

/*Routine for writing into a file from a buffer*/
file_write()
{ byte2=write(hand2,obuffer,j);
  if (byte2==-1) printf("\n unsuccessful write");
}

/*Routine for whole operation to work*/
main()
{ int k=0;
  while (hand1==-1) file_open();
  while (hand2==-1) file_creat();
  sendvt(); ++j; obuffer[ij]=0x0c;
}

```

```

while (size1!=0)
{ file_read();while (i<size1)
    {if (row==0) {obuffer[j++]=0x0b;while (he[k]!=0)
        obuffer[j++]=he[k++];obuffer[j++]=0x0b;row=1;k=0;}
    else if (row<=text ) {d=ibuffer[i];d=d & 0x7f;
        conversion();}
    else {obuffer[j++]=0x0b;while (fo[k]!=0)
        obuffer[j++]=fo[k++];obuffer[j++]=0x0c;
        row=0;k=0;col=1;++page;} }
    file_write();i=0;j=0;fp1=fp1+size1;fp1=lseek(hand1,fp1,0);
    fp2=fp2+byte2;fp2=lseek(hand2,fp2,0);printf("\n %d",fp2);
}
hand3=close(hand1);hand4=close(hand2);
if (hand3!=-1 || hand4!=-1) printf("\n %d, %d",hand3,hand4);
}

```

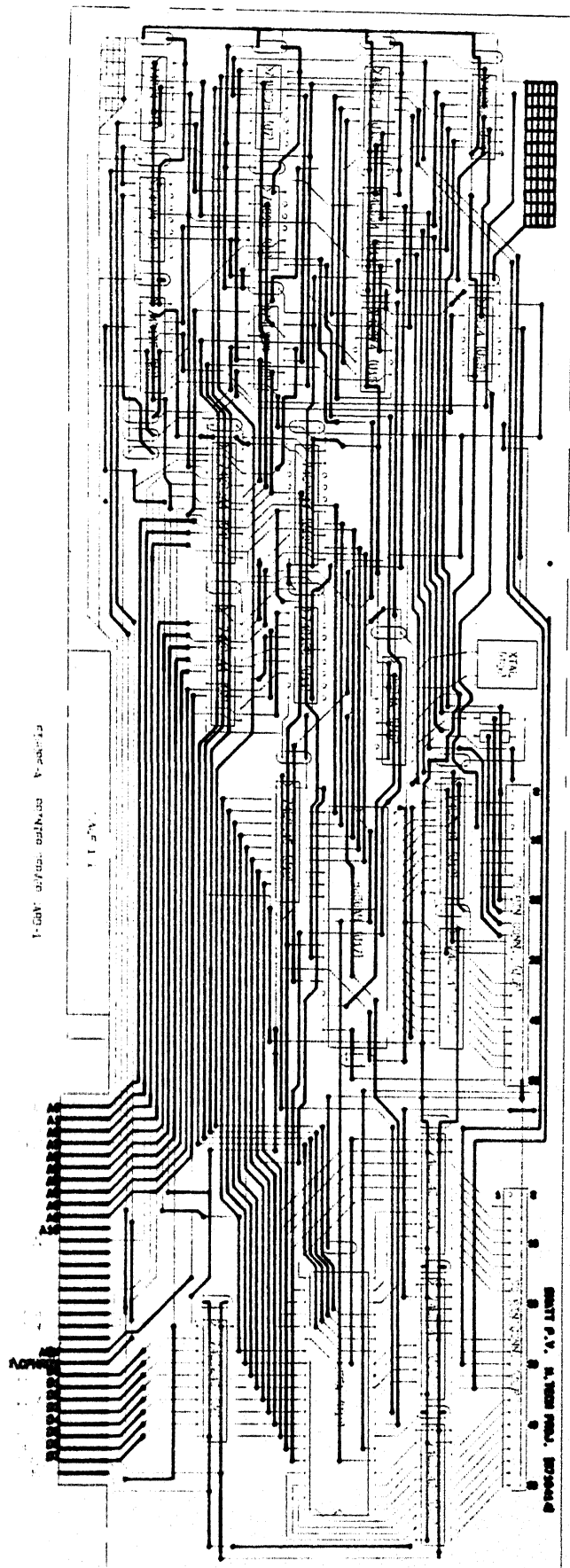


FIG-2 : PRINTER SERVER CARD-1

FIGURE 3: PRINTER SERVER CARD-2

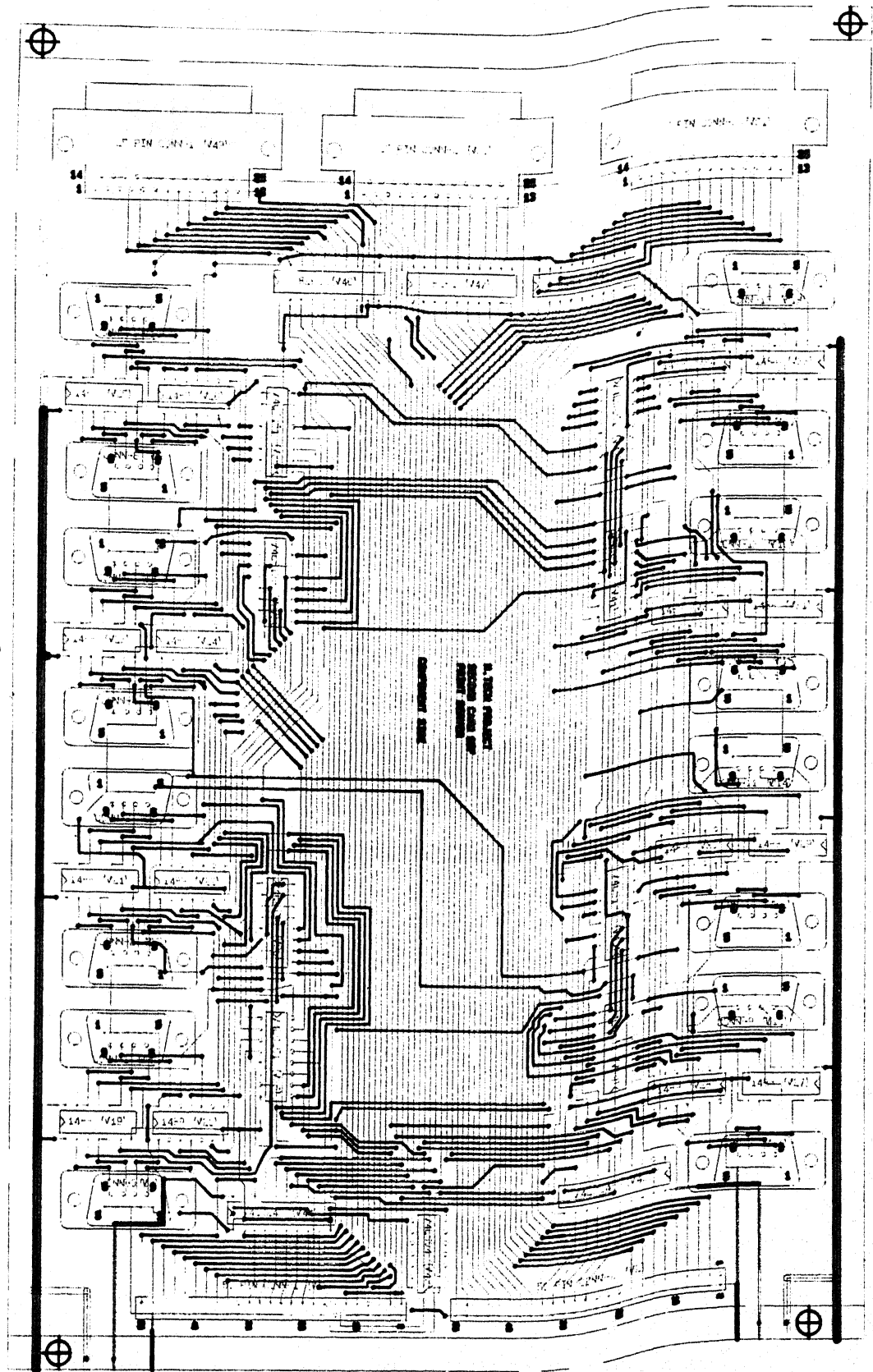
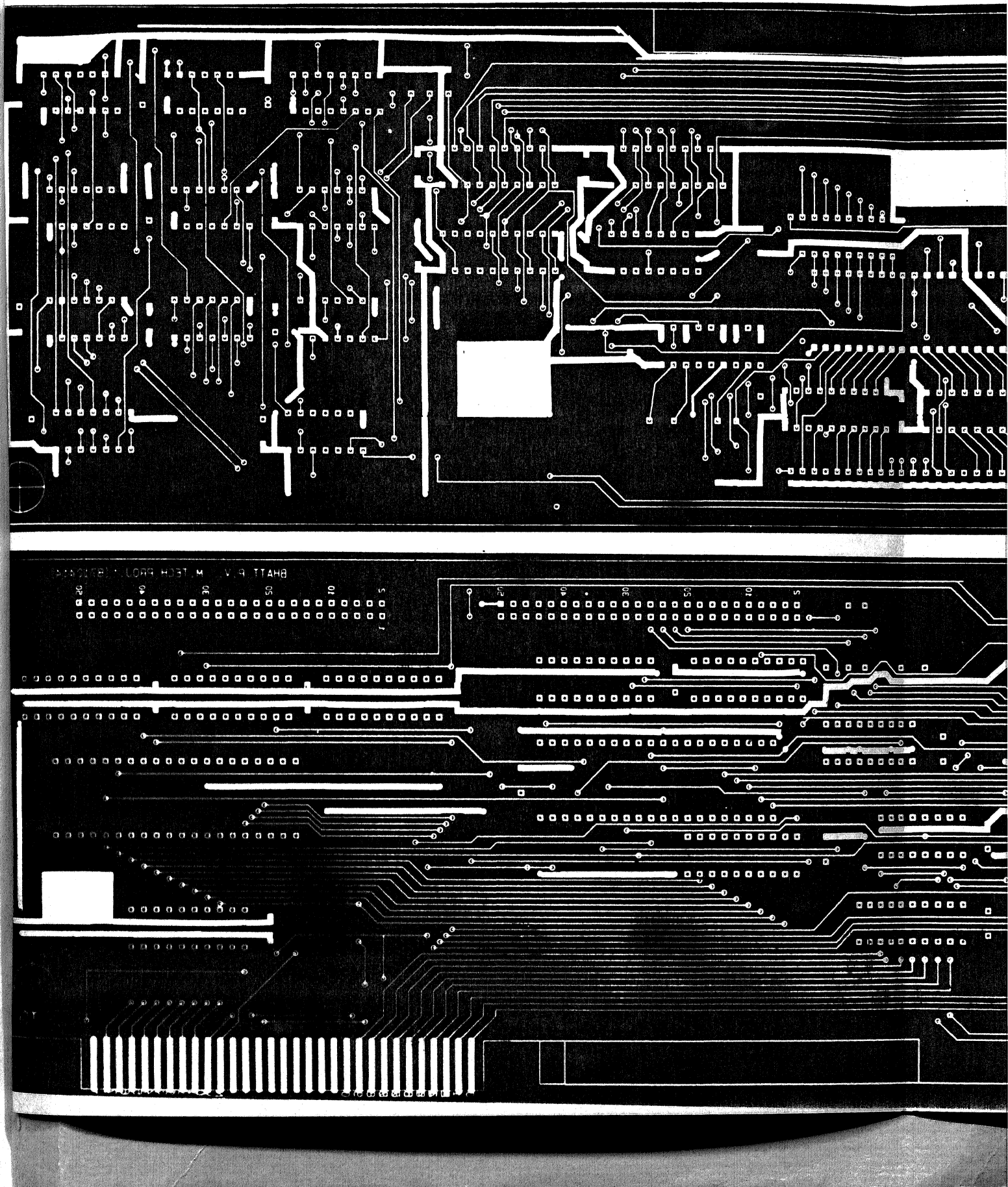
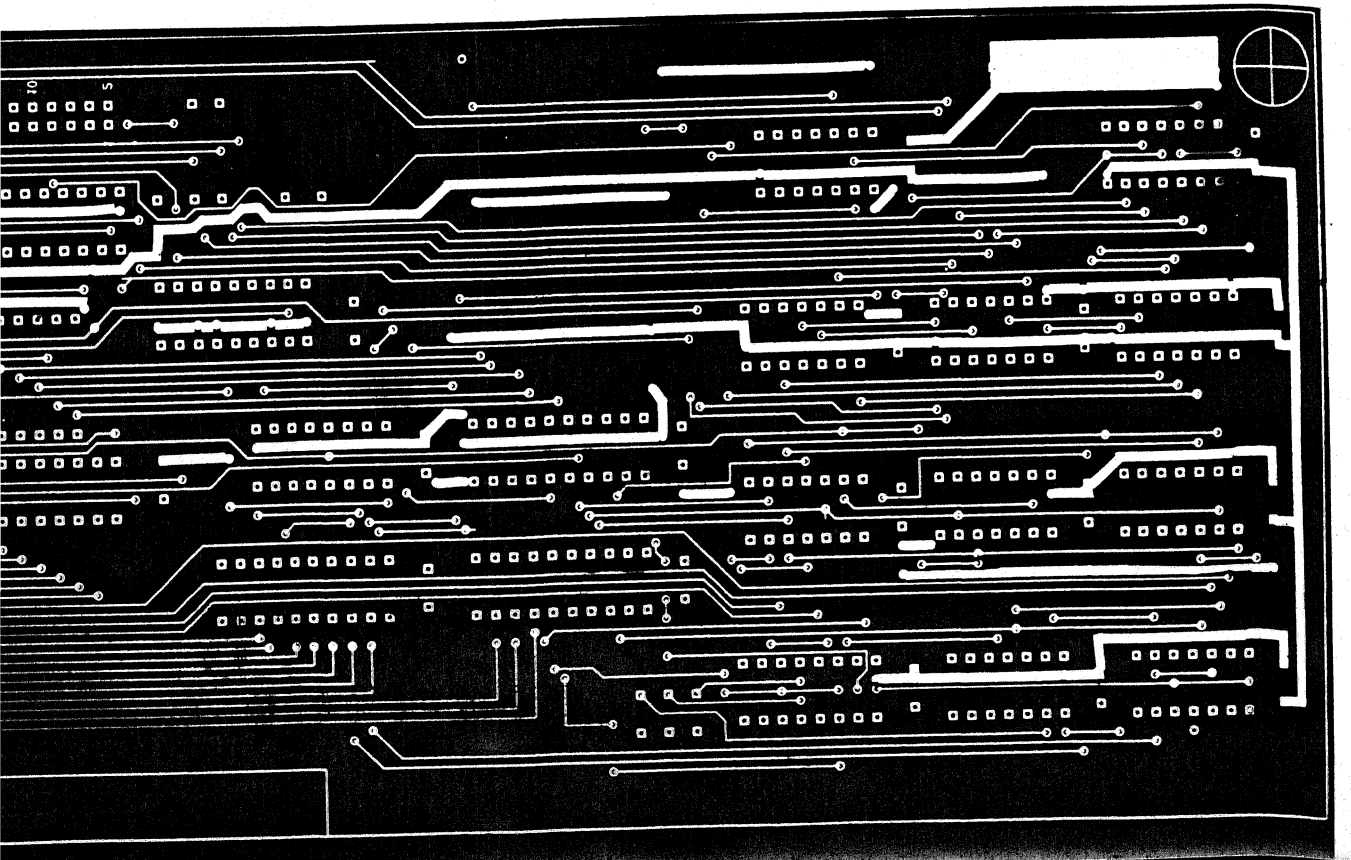
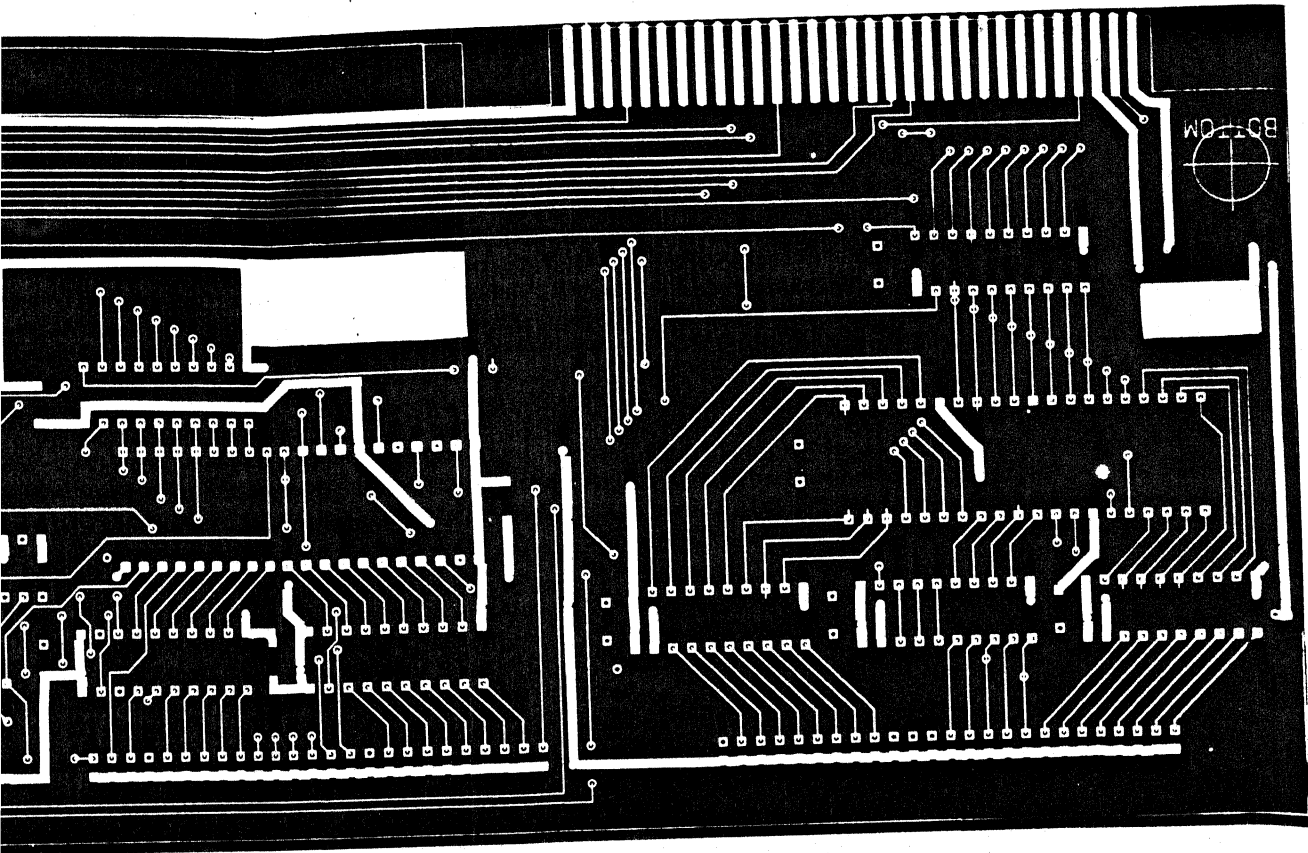


FIG-3: PRINTER SERVER CARD-2

FIG-4 : NEGATIVE PHOTOCOPY OF PRINTER SERVER CARD-1 PCB





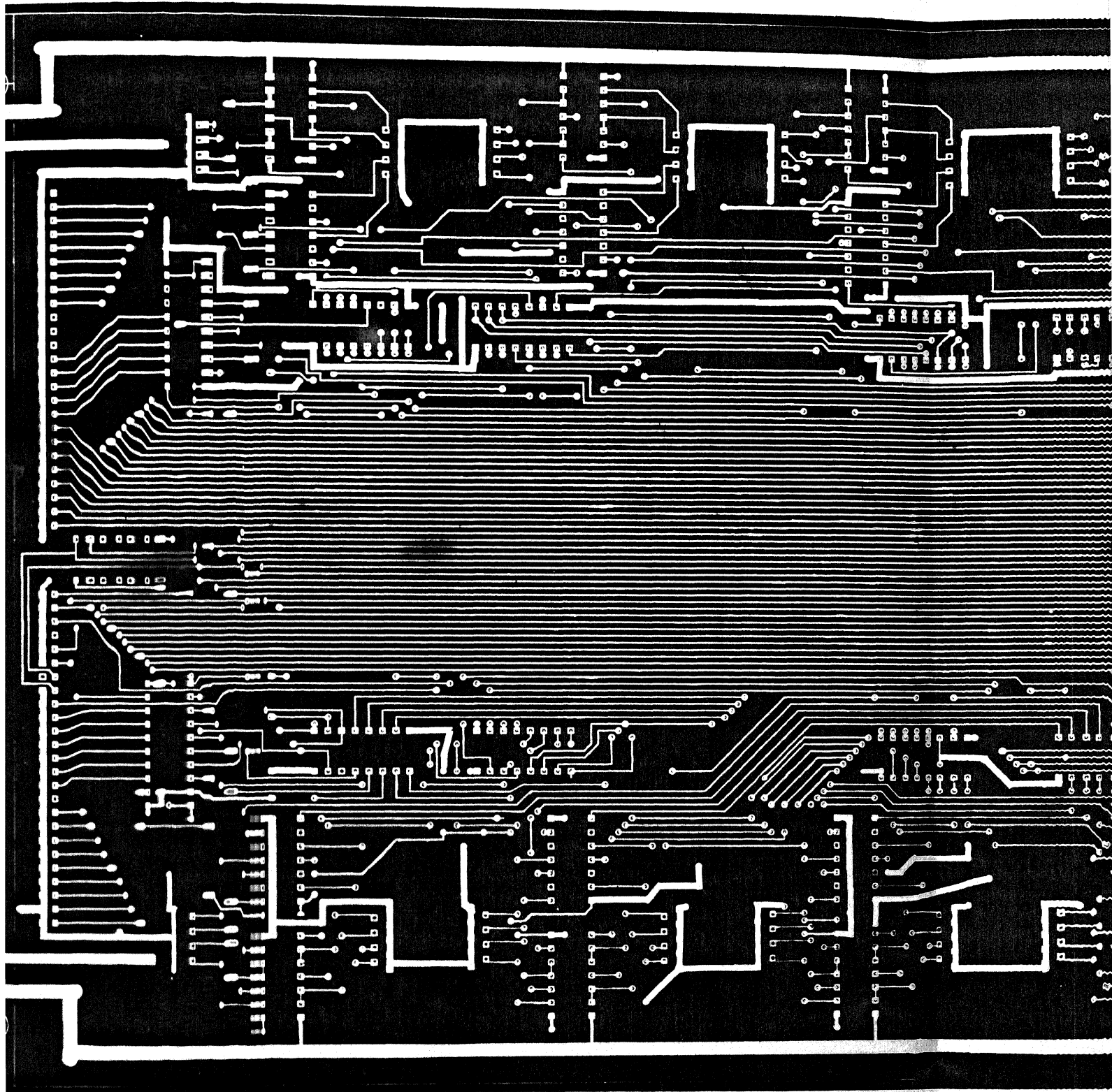
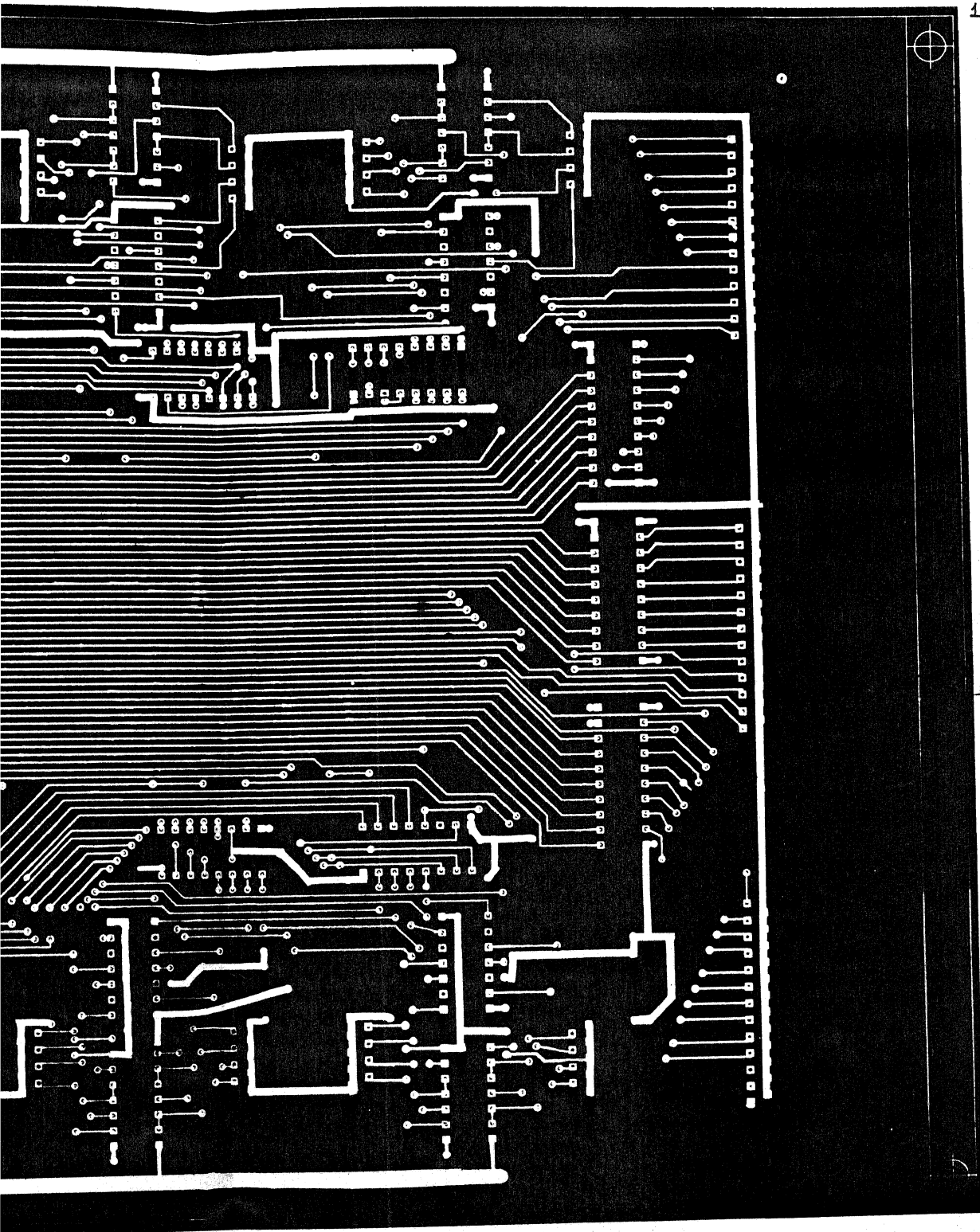


FIG- 5 : NEGATIVE PHOTOCOPY OF PRINTER SERVER CARD-2



OF PRINTER SERVER CARD-2 PCB [SOLDER LAYER]

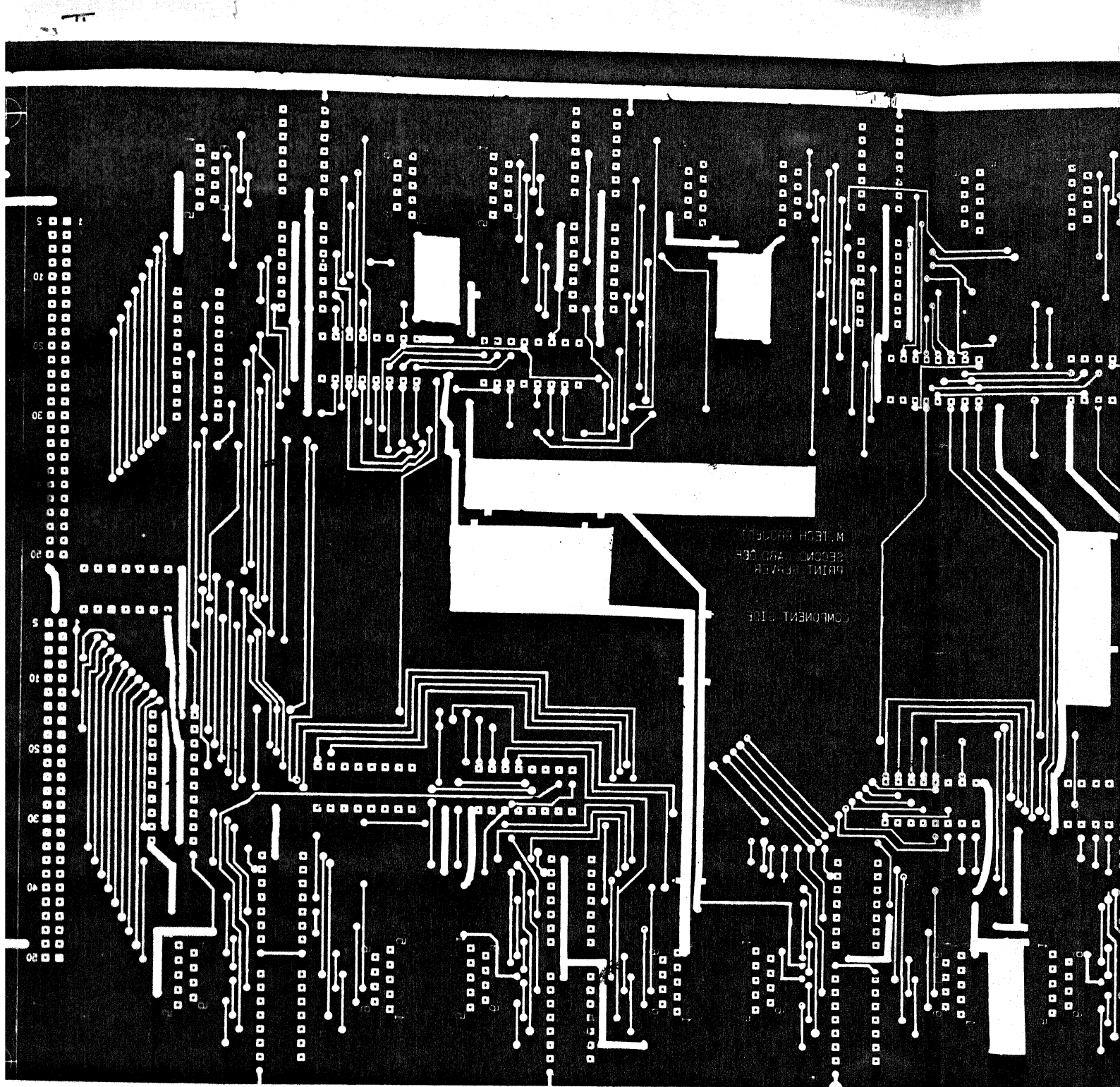
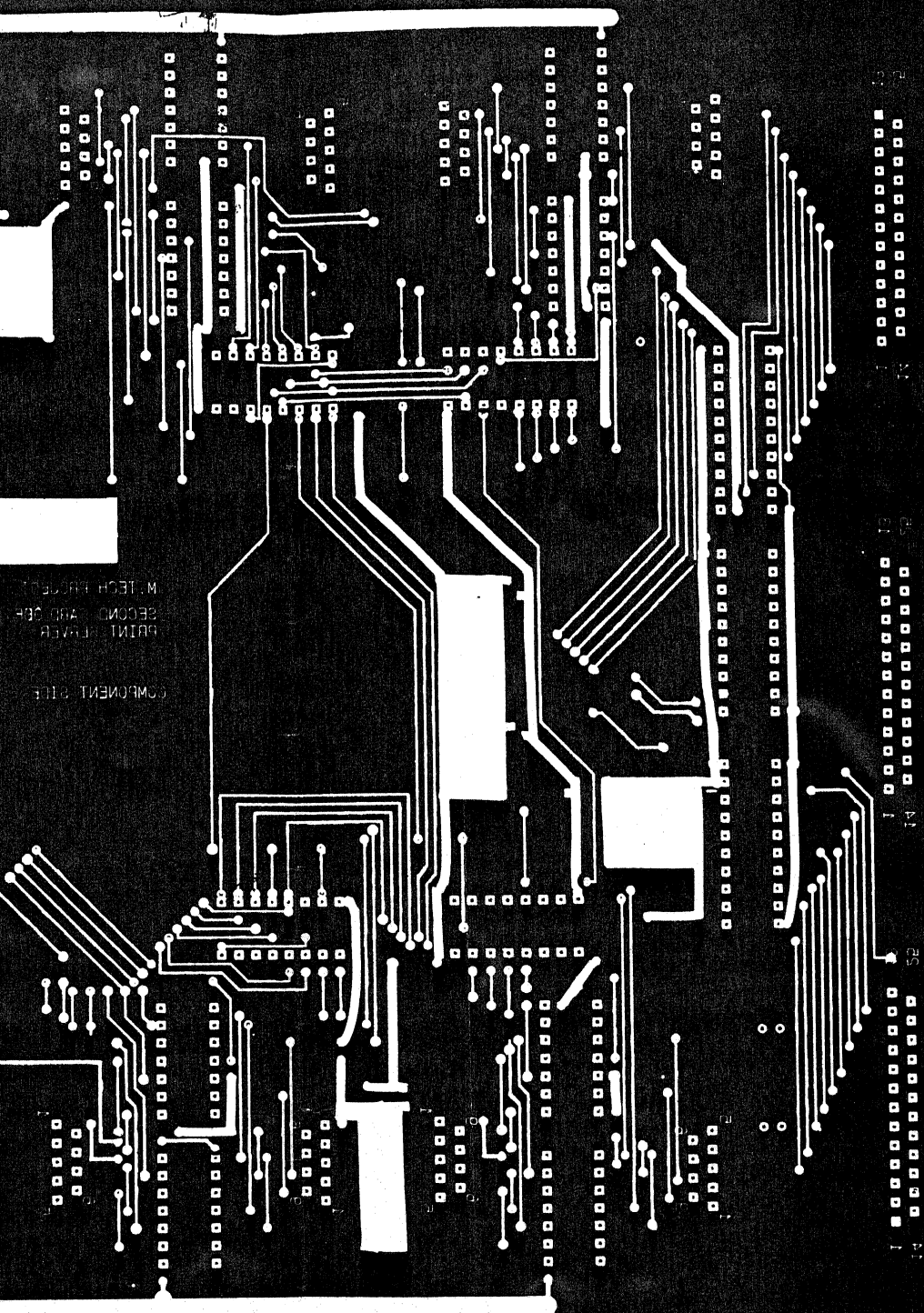


FIG-6: NEGATIVE PHOTOCOPY OF PRINTER SERVER CARD-2



OF PRINTER SERVER CARD-2 PCB [COMPONENT LAYER]

105876

Th
621.38195 Date Slip 105876

B4.69d This book is to be returned on the
date last stamped.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

EE-1909-M-BHA-DES